Vulnerability management service for product life cycle

Andon Nikolov

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 22.5.2017

Thesis supervisor:

Prof. Heikki Hämmäinen

Thesis advisor:

M.Sc. Matti Frisk

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Andon Nikolov

Title: Vulnerability management service for product life cycle

Date: 22.5.2017 Language: English Number of pages: 7+66

Department of Communications and Networking

Professorship:

Supervisor: Prof. Heikki Hämmäinen

Advisor: M.Sc. Matti Frisk

This thesis was commissioned by a large enterprise. The company requires a vulnerability management solution, which would enable them to manage vulnerabilities throughout the product life cycle. An analysis was required on whether such solution should be purchased or built as an internal project.
This study was completed in two main phases. First, a make-or-buy decision was done based on the analysis. Second, a suitable VMS design and implementation was suggested.
To collect input for the analysis, all potential users were identified and from them groups of volunteers were invited to interviews. The data from the focus group interviews was then processed and documented in the form of requirement specification for Vulnerability Management Service (VMS). Commercial off-the-shelf solutions were compared against the list of requirements. A second round of review was done with selected commercial products, which fulfilled majority of the requirements.
As a result of the performed comparisons, this study concluded that building an own solution would deliver higher Return on Investment (ROI) in long term perspective. VMS stakeholders accepted the recommendation of this study and proceeded to fund the design and implementation.
The study goes on to provide guidelines for service design and implementation based on industry best practices. This paper also introduces a useful maturity model for VMS capabilities and monitoring of the evolution of vulnerability management practices.

Keywords: Vulnerability, management, service, product, life cycle, VMS, PLCM

Preface

I want to thank Professor Heikki Hämmäinen and my instructor Matti Frisk for their great guidance and support. Without their help, this thesis would have never been completed!

Otaniemi, 22.5.2017

Andon Nikolov

Contents

Abbreviations

2PP	Second Party Product (product/component developed by the same company which is used as part of the product in question)
3PP	Third Party Product
AD	Active Directory
API	Application Programming Interface
BCM	Business Continuity Management
CAPEX	Capital Expediture
CEO	Chief Executive Officer
CERT	Computer Emergency Response Team
CIS	Center for Internet Security
COTS	Commercial Off-The-Shelf
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVRF	Common Vulnerability Reporting Framework
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DDoS	Distributed Denial of Service
DNS	Domain Name System
DRP	Disaster Recovery Planning
EVMS	Enterprise Vulnerability Management Service
EU	European Union
FIPS	Fedaral Information Processing Standard
FOSS	Free and Open Source Software
HBAC	Host Based Access Control
HIPAA	Health Insurance Portability and Accountability
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol (RFC 2616)
IaaS	Infrastructure-as-a-Service
ICT	Information and Communications Technology
ICASI	The Industry Consortium for Advancement of Security on the Internet
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol (RFC 791)
IPR	Intellectual Property Rights
IPS	Intrusion Prevention System
ISMS	Information Security Management System
ISO	International Organization for Standardization
ISP	Internet Service Provider
IT	Information Technology
Mbps	Megabits per second
MVP	Minimum viable product

NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OASIS	Organization for the Advancement of Structured Information Standards
OpenPGP	Open Pretty Good Privacy
OPEX	Operational Expenditure
OS	Operating System
OVAL	Open Vulnerability and Assessment Language
OWASP	Open Web Application Security Project
PCI DSS	Payment Card Industry Data Security Standard
PDU	Product Development Unit
PLCM	Product Life Cycle Management
PSIRT	Product Security Incident Response Team
PST	Product Security Team
RA	Risk Assessment
RBAC	Role Based Access Control
REST	Representational state transfer
RFC	Request for Comments
ROI	Return on Investment
ROSI	Return on Security Investment
RQ	Requirement
RSS	Really Simple Syndication
QoE	Quality of Experience
QoS	Quality of Service
SaaS	Software-as-a-Service
SCAP	Security Content Automation Protocol
SDDC	Software-Defined Data Center
SDN	Software-Defined Networking
SIEM	Security information and event management
SLA	Service Level Agreement
SOC	Statement of Compliance
SSH	Secure SHell
S/MIME	Secure/Multipurpose Internet Mail Extensions
TLS	Transport Layer Security (RFC 5246)
UI	User Interface
URI	Uniform Resource Identifier
USM	Unified Security Management
UX	User Experience
VA	Vulnerability Assessment
VM	Vulnerability Management
VIM	Vulnerability Intelligence Manager
VMS	Vulnerability Management Service
WiFi	Wireless local area network trademark of the Wi-Fi Alliance used to certify compatible devices

1 Introduction

1.1 Background

Software has growing importance in our daily life. To increase efficiency in the development of software, products and services, many companies are using software components developed by third parties (3PP). Many of these components are often Free Open-Source Software (FOSS). With the increasing utilization of such popular components, the impact of the vulnerability on company assets becomes increasingly dramatic. An example of such cases are the Heartbleed and Shellshock incidents, where vulnerabilities in the popular OpenSSL suite and Bash shell rendered millions of Linux systems vulnerable. For many small companies it could be a matter of hours to update and patch their software or service. On the other hand, the problem created by such vulnerability is many orders of magnitude higher for large enterprises with hundreds of products and services.

This problem is further escalated by the rapid increase in Internet of Things (IoT) deployments. During 2016 new record breaking DDoS attacks were launched against some of the biggest hosting providers [33], [30]. Those attacks were attributed to new botnet called Mirai, which was composed primarily of IoT devices which had factory settings, or had software with known vulnerabilities [29], [69].

When one considers the sheer volume of code that is produced in this day and age, even small probability of error becomes a problem. Based on industry average, there could be 15 to 50 errors in 1000 lines of code. In cleanroom development it can be lowered to 0.1 errors in 1000 lines of code. That said, if a product has over 1 million lines of code, statistically it will have between 100 and 50 000 flaws that could turn to vulnerabilities [3] [31]. It is clear that mistakes are bound to happen all the time, they just have to be handled swiftly to minimize impact of the vulnerability.

To be able to react in timely manner and reduce the risk from vulnerable products and services, large enterprises need tools and processes to support their product development and maintenance activities. Vulnerability Management has been defined as "cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities" (As cited in [11], Cornell 2009). Highest priority is given to mitigating the risk in currently deployed products, especially if they are Internet facing. Eventually, mitigation is applied to products that are still under development in order to guarantee that products will leave the company vulnerability free.

It is very important to understand that implementing any process in enterprise environment takes time. It is expected that enterprise wide processes and systems are deployed in steps, often referred to as maturity levels. Thus, when implementing vulnerability management at enterprise scale, it will follow similar pattern and evolve through number of levels of maturity.

1.2 Research question and scope

1.2.1 Research question

Most of the vulnerability management efforts are focused on detection and mitigation of vulnerabilities in already deployed products and services. It is clear that majority of the Vulnerability Management Service (VMS) will be focused on the same area, thus leaving vulnerability management for product life cycle as an after thought. The fact that (at the time of writing) Google Scholar is able to find 5300 results for "vulnerability management", 318 for "vulnerability management system" and only 34 results with reference to "vulnerability management service" is an indication of how under-studied the topic is.

There are two main use cases for vulnerability management discussed in this thesis: IT operations and product development. In IT operations, there are systems and infrastructure which are utilized to deliver a service. The software and hardware used, are not created by the IT operations team. Thus, in IT operations context, vulnerability management is focusing on detecting vulnerabilities and mitigating them. Both actions are performed in the same team and there is minimal need for external communication. In contrast, vulnerability management in product development is more complex process. Vulnerabilities are being detected in software which used or is still under development. The organization has responsibility to mitigate all vulnerabilities found in their products and communicate to their customer the impact of all found vulnerabilities.

Vulnerability management process can be implemented using vulnerability management system, which in turn can deliver VMS to variety of users. VMS are used to enable timely actions, notify all affected product and service organizations, and to track the progress of mitigation steps. VMS is characterized by improved efficiency in vulnerability management and mitigation, through automation and process enforcement. VMS are often used in Information Technology (IT) operations environments. In that context VMS includes: scanners to identify new vulnerabilities, back-end processing for classification and remediation proposal, front-end for visualization of the results and follow-up process to apply and verify the mitigation.

While such VMSs are useful for computer networks and systems, they are not so effective in software development context. The products under development have not yet been deployed, which renders VMS scans useless. This creates a need for different type of detection process, based on heuristics and product specific information. IT centric VMS will be able to classify and suggest remediation, once a vulnerability has been identified. In addition, in product development it is important to distinguish between vulnerabilities which have impact on the product and vulnerabilities which can lead to exploitation of the product. When vulnerability is detected in a product, if the attack vector is not available in this particular product, it makes the product impacted by the vulnerability, yet not vulnerable. This brings us to the last part of the process: the mitigation. It is non-trivial to verify vulnerability mitigation in IT system. In product development, such test becomes a real challenge. Stemming from the fact that the product is not deployed, the techniques that can be used to confirm the mitigation are very limited. In most cases this challenge is handled by

complex software security assurance processes.

Furthermore, as every large software house has its own internal ways of working, it would be difficult to create a standardized, common VMS for all use cases. Although many companies offer VMS services [cit.], these have had only limited success. Some of the reasons for this include the confidential nature of the processed information, with few companies willing to share this information with external parties; the need for integrity protection; expectations concerning the availability of the service; and integration with existing infrastructure.

Unfortunately, these requirements cannot be fulfilled using current off-the-shelf VMS products. Therefore, the objective of this thesis is to design an architecture for a VMS to be incorporated into a software development and maintenance life cycle. To accomplish this task, the thesis will compare a number of existing VMSs and evaluate their usefulness in terms of confidentiality, integrity, availability and usability of the service.

Research question: Which solution is able to deliver enterprise scale, cost effective and fast vulnerability management service for product life cycle?

1.2.2 Scope

This thesis will focus on the vulnerability management practices in companies that develop software or products which run software. Some of the practices shown here may apply to other industries as well, however, that is a topic for a separate research. To be able to deploy VMS, the organization must start by defining its vulnerability management vision, devise vulnerability management strategy and create vulnerability management policy. If these prerequisites are not met, VMS cannot be effectively deployed.

The scope of the study does not cover the definition of vulnerability management vision, strategy, policy and processes.

As a part of this study, the author analyzes vulnerability management policy requirements, collects user expectations, refines all stakeholder input and proposes a solution, which can fulfill all user requirements. In addition, the author has contributions for definition of vulnerability management process and has provided assistance during service design, implementation, testing and deployment phases.

The goals of this study are:

- Produce detailed requirement specification
- Analyze Commercial Off-The-Shelf (COTS) products which may fulfill the requirements for VMS
- Provide input for make-or-buy decision
- Support service development and implementation

In addition, this thesis covers the importance of definitions, classification and standards. In the context of this study, it is valuable to define: vulnerability, impact, severity and priority. The study also covers classification and scoring of vulnerabilities.

The most relevant standards for this thesis are ISO/IEC 29147 [25], ISO/IEC 30111 [24] and National Institute of Standards and Technology (NIST) originating: Common Platform Enumeration (CPE) [35]; Common Vulnerabilities and Exposures (CVE) [36]; Common Vulnerability Reporting Framework (CVRF) [23]; Common Vulnerability Scoring System (CVSS) [12]; Common Weakness Enumeration (CWE) [34]; Security Content Automation Protocol (SCAP) [42]; Open Vulnerability and Assessment Language (OVAL) [37];

1.3 Research methods

This study employs qualitative research strategy, as it fits better the research topic. The study utilizes focus group interviews and make-or-buy decision analysis. The interviews facilitate the collection of user requirements, while the make-or-buy analysis provides input to stakeholders for decision. Furthermore, this study compares COTS products and evaluates what portion of the specified requirements could be fulfilled. The results of the evaluation are used as an input for the make-or-buy decision.

1.4 Thesis structure

The rest of the thesis is divided into four chapters. Chapter 2 will present the state of the art in the field of vulnerability management by reviewing popular literature on the topic. Chapter 3 will provide an analysis of the requirements for VMS for a complete software Product Life Cycle Management (PLCM). Chapter 4 will discuss the proposed solutions and provide benchmarks (compliance to the requirements). Finally, Chapter 5 will reveal the conclusion of the thesis, summarize the findings and propose areas for future research.

2 Literature review

2.1 Importance of security

Every year, there is an increasing amount of news on cybersecurity, data breaches, user data leaks, confidentiality and privacy violations. For example during 2016 we have seen [59]:

- Yahoo - 1 billion users targeted, which broke their previous disclosure record of 500 million accounts earlier the same year
- LinkedIn - reports 117 million usernames and passwords stolen back in June 2012.
- AdultFriendFinder - 412 million usernames and passwords stolen.
- Government - various embarrassing leaks related to parties and president elections.
- US NSA stolen tools being leaked
- Home WiFi routers by Netgear and D-Link being vulnerable and allowing complete remote access.
- Various devices with backdoors. Mostly cheap Android phones, but some laptops too.
- Bangladesh Central Bank robbery ($101 million)
- Ransomware attacks
- Malvertizing (Malicious Advertising) campaigns.
- Mirai and other IoT botnets
- Traditional Adobe Flash player vulnerabilities

Recently, a new ransomware called "WannaCry" has emerged and in less than two week has taken over 200 000 machines [61], [60]. Some of the impacted systems belong to governmental organizations and enterprises. This particular malware is exploiting known vulnerability in Microsoft Windows Operating Systems (OS), for which fix has been available over a month. This example highlights the bad security and patching practices in the IT industry. It also highlights the negative impact of state organizations' stockpiling of vulnerabilities, which were leaked on the Internet. It is important to notice that this news was popular with the general audience, and not only with security researchers and professionals. Some of the items listed above made it to top of 2016 technology failures overall too [65], [22], [28], [50].

Having in mind how much technology has permeated our daily life and how much of the technology depends on some kind of software or firmware, it is only natural that software weaknesses would have increasing impact on us. A software flaw in

life support device might even cost a human life. Even cars that are dependent on software for features as autopilot have already caused human casualties [77]. Cases where hackers would remotely control victim's car from science fiction just became scary reality [18]. Such facts should make people aware of the need for better software security and improvement in the handling of software flaws.

To be able to understand the basis of information security, it is recommended to start with definitions. An informative summary on the topic has been written by Andersson [1]. Starting with the CIA model - Confidentiality, Integrity and Availability, is widely used and easy to comprehend framework for information security. As suggested by Andersson, to measure the impact, one should measure the elements of CIA, a measurement which is hard to perform due to lack of standard metrics.

Most often the metric used is raw number of incidents and the cost of compromise. With the exception of HIPAA [71], FIPS [40] and PCI DSS [52] the IT industry has not been regulated. The HIPAA, FIPS and PCI DSS compliance are the widely used requirements in the IT world due to the fact that solutions, which should handle health, governmental and payment data are regulated and compliance to the requirements can be enforced by governments and regulator.

There is a number of other requirements applicable for specific countries (India, China) or international entities (EU), yet those requirements are bound to their localities and can be enforced only within their borders. This further strengthens the point made by Andersson for a lack of general metrics for information security.

Furthermore, there is a lack of widely accepted definition for information security. The argument continues describing the difficulties of evaluating risks and their probabilities especially when those risks originate in human behavior. The same argument is made also by Hall [19], who reminds that a system is only as secure as its weakest link and employees are rarely motivated to maintain strong system security. Hall goes on to, emphasize another well know problem: companies measure their employees' performance based on sales targets, deadline fulfillment and costs saved.

This industry practice leads to a model where a company will always prefer to deliver a product or service on time at the expense of security. Having a limited budget management, will often prioritize feature development (which is easy to measure), compared to security controls, which might even disable some wanted product functionality. In addition, if a new product has not been compromised, it is hard to justify if it is due to proper security or due to the fact that nobody has attacked it yet.

However, as Andersson suggests, this is not a reason to give up and abandon security. He proposes that information security in enterprise should be "A well-informed sense of assurance that information risks and controls are in balance." This concept could be defined as business optimal security, which implies that there should be balance between the business objectives and the security measures implemented. Furthermore, such assurance helps to define metrics for information security, which in turn makes it easier to implement and monitor. Assurance is also something that can be communicated to customers, thus increasing the trust and visibility on

enterprise security practices.

2.2 State of computer security

The Economist has provided an easy to follow summary on the current state of computer security [3]. The author shows how the problems we have seen in recent years are not exceptions, but rather consequence of outdated way of thinking. The analysis also includes valuable insight, suggesting that even from probabilistic point of view, the software that we use today is bound to have flaws. Considering that the 2016 model Ford F-150 pickup truck has over 150 million lines of code [9], and as pointed out by the Economist even the best software houses would have 0,5 error in 1000 lines of code, theoretically, the truck would likely have 75 000 errors in its code.

It is a well known fact that an attacker needs to find a single weakness to compromise a system, while the defenders must prevent exploitation of all known and unknown vulnerabilities. The Economist continues, discussing that all software is vulnerable, and also confirming that the reasons are software development culture, rapid business growth and economic incentives. In addition to the points brought by Hall, the author brings in the incentive of governments to reduce computer security, thus, enhancing their spying capabilities.

However, these are challenges that can be tackled. For example, it is possible to create mission critical software with minimum flaws by using formal methods approach [2]. Formal methods allow to mathematically prove that software will function only as intended, thus producing software without vulnerabilities. These development methods however, come at a high cost. They require highly skilled professionals with development and mathematical skills. The process is longer, due to the strict requirements needed to be fulfilled. Due to the high cost, formal methods development is only suitable for mission critical software, where software quality is prioritized over cost.

This brings us again to the concept of business optimal security. Companies have to accept that their software will have vulnerabilities, and develop processes to mitigate the problems in a timely manner. From users' perspective, the challenge is to implement enough security controls and protection to make the cost to exploit the potential flaws higher than the benefit for an attacker. As we have seen, ROSI (a form of ROI) is hard to define and measure, thus for-profit organization have to justify and spend limited financial resources in the most business beneficial way to protect their assets.

2.3 Security threat intelligence

It is important for a company to identify their risks and where security budget should be spent. One way is by performing Risk Assessments. A common way is to use the ISO/IEC 27000 family of standards for reference in such assessments [26]. Threat intelligence reports can be used to better evaluate the risk severity and probability [32], [33], [64], [68], [67]. The facts in afore mentioned reports show that computer threats are growing continuously. Each year there is growth in:

- Security breaches
- Identity theft
- Malware variants (added each year)
- Crypto-Ransomware
- Mobile platform and software vulnerabilities
- Web attacks
- Zero day vulnerabilities

All these trends are further accelerated by growth in IoT. As the market demand for cheaper connected devices continues to be strong, vendors have incentive to fulfill the need by cutting cost from non-essential components. Unfortunately, one such non-essential component happens to be security. With more devices, which are able to run code and are connected to our home networks or the Internet, our risk exposure is increasing at high rate.

2.4 Software faults and vulnerabilities

There are variety of software flaws. When a product is not functioning as expected or specified, the flaw is considered a bug. It is something to be corrected, to make the product behave as expected. When a software flaw could leads to product exploitation by malicious actors, it is considered vulnerability. In this thesis, the distinction be bug and vulnerability is important due to the fact that the process to correct it can vary. It is relevant to note that while bug can be found with rigorous testing of expected functionality, vulnerabilities are often found with negative test cases. Negative test cases are such in which the product is tested in ways in which it is not expected to be used. While, it is possible to automate testing processes and test with positive cases, to make sure that all expected functionality is in place, negative test case can be considered infinite for all practical purposes. Figure 1 and Figure2 visually represent the processes for software bug and vulnerability handling respectively.

Figure 1: Bug handling process

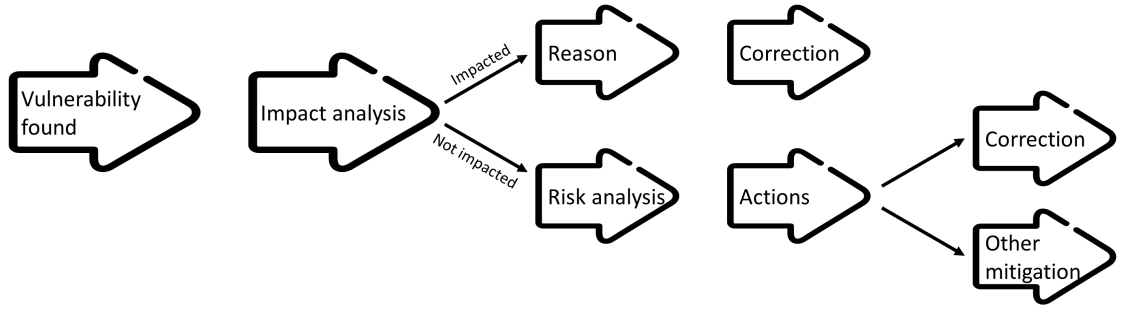

Figure 2: Vulnerability handling process

2.5 Security statistics

Another valuable source of security information are vulnerability and 3PP databases. Starting with the de-facto standard National Vulnerability Database (NVD), let us have a look at the amount of new software entries that are registered in the CPE dictionary every year. The CPE dictionary, maintained by NVD, is industry standard source of 3PP information. CPE entries have standard name formatting, which makes it useful for product vendors to communicate in clear and standard way to their customers. It is also useful for automated processing of vulnerability information.

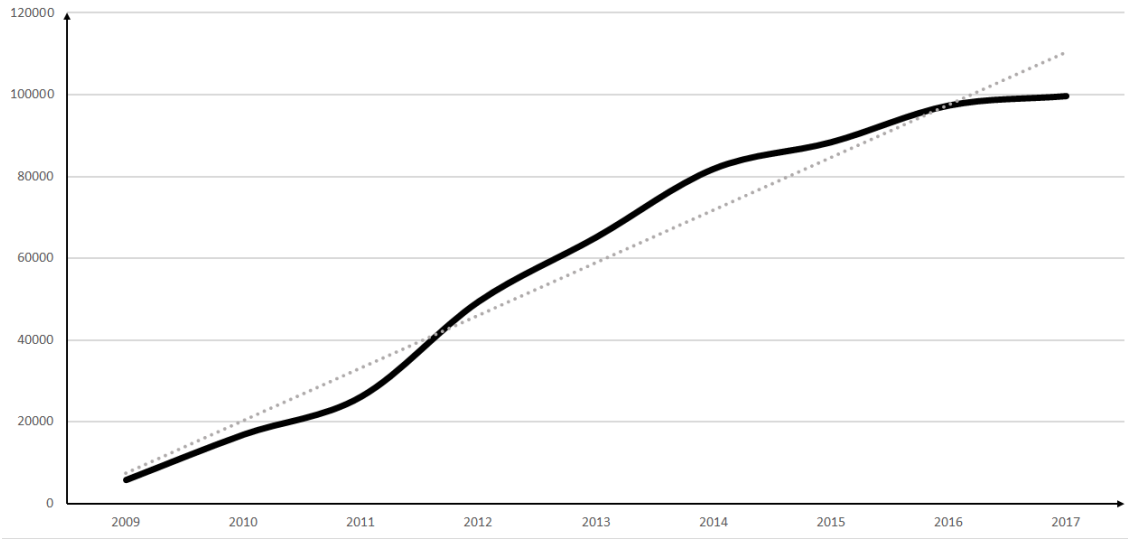

Figure 3: New entries registered in CPE dictionary [44]

From Figure 3 it can be observed that there is clear, increasing trend. Using linear approximation, the estimated growth rate is 12883 new entries per year. Considering this rate, even with improvements in software quality, it is not likely that overall vulnerability trends will decrease. [44]

To supplement the new CPE entries data, attached were also the statistics for new, published by NVD, vulnerabilities. Note that there is a larger number of vulnerabilities with reserved CVE identifiers, which have not been published yet.

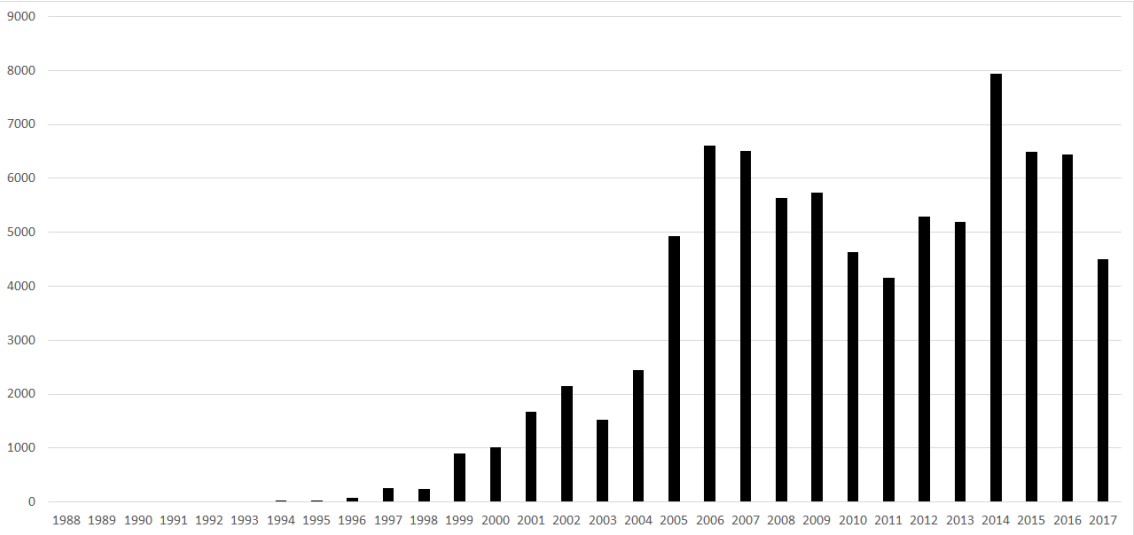

Figure 4: New vulnerabilities in NVD [45]

The data in Figure 4 spans over longer period of time to better visualize the fluctuations in amount of vulnerabilities published. The figure indicates that there was an increasing trend until 2006, which then declined with local minimum in 2011. Then a second increase started with a peak in 2014.

2014 was the year of the so-called "mega-vulnerabilities": Shellshock, Heartbleed, POODLE. Those vulnerabilities had such impact due to the fact that they were found in widely used components (BASH, OpenSSL). As BASH and OpenSSL were present in the majority of Internet connected servers, the threat and risk of exploiting these vulnerabilities was high, which in turn explains the unrivaled before media coverage.

In 2015 and 2016, there is again decline. While there is no clear indication of the reason for this decline, a number of factors could contribute to it. Difference in counting methods for vulnerabilities is one such factor. Some researches and companies would bundle multiple related flaws in one CVE identifier, while others will break it into multiple CVE identifiers. Also, the statistics only accounts for published in NVD vulnerabilities, a number that could be different from total amount of found vulnerabilities. That said, 2017 is actually on track to become a new peak with projection of over 10000 vulnerabilities published. On average, NVD has 2816 vulnerabilities recorded for the last 30 years. If the focus is on the last 10 years, the average is more than double at 5801 vulnerabilities per year [45]. In comparison, commercial VulnDB statistics show even more vulnerabilities registered in their private database every year.

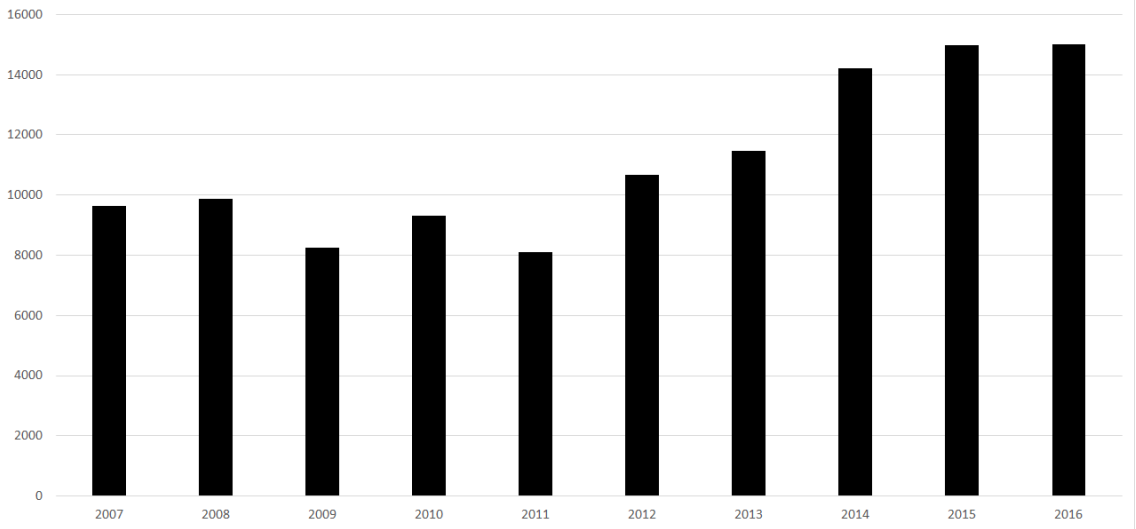

Figure 5: New vulnerabilities in VulnDB [74]

As mentioned earlier, NVD is the de-facto standard, but there is no enforcement process to ensure that all vulnerabilities are tracked in NVD. As a result, commercial vulnerability databases can include greater number of vulnerabilities, as the company operating the database has commercial incentive to collect more data. In addition, there are large number of vulnerabilities in NVD which have CVE identifiers reserved, but are not published, thus, not counted in the NVD statistics above [74].

Similar results are seen also from Flexera's vulnerability database, where 17147 vulnerabilities were recorded in 2016 [14]. Their report also suggests that as vendors and amount of vulnerable products in their database has decreased, with 7% and 14% respectively, the number of vulnerabilities per product has increased.

Having discussed the current state of cyber security and the trends in vulnerabilities, the next sub-section will cover the methods used in this thesis.

2.6 Methodology

The following part will present the methodological choices and considerations for this thesis, as well as the decisions related to what has been studied, which approaches have been used, and how it was conducted. This is done in order to create transparency and to provide the reader with a better understanding of the results.

A major decision that has to be made about every study is the general orientation to the conduct of the research. There are two distinctive clusters of research strategies: quantitative and qualitative, that have different foundations. Quantitative strategy emphasizes the quantification in the collection and analysis of data, whereas qualitative strategy focuses on gaining an understanding of underlying reasons, opinions and motivations.

The focus of the present paper is placed on the service users in their various roles and their understanding of the problem, as well as their vision as to how it can be solved most efficiently. Therefore, the qualitative strategy is better suited for the

purposes of the present thesis. This strategy allows going in-depth with the available empirical data and gaining a better understanding of the studied matter.

2.6.1 Focus group

The research design employed in this study is known under the name focus group. As Bryman [5] explains, this particular design could be very helpful in the "elicitation of a wide variety of different views in relation to a specific issue". It involves two or more interviewees at the same time, who engage in a discussion about the studied topic. During this discussion, they are likely to hear new opinions from each other and to get to a new, deeper understanding of the studied topic. Also, they might challenge each other's views, thus coming up with new solutions of the problem at hand.

For the purposes of this paper, the focus groups entailed semi-structured interviews on a specific topic with up to 10 participants at the same time. This ensured that the group is big enough but not to the point where it is unwieldy. The selection of participants was done in two stages. First, seven categories of service users were identified and then from each category volunteers were invited to take part in the study. This was done in order to ensure more productive interviews, where participants demonstrated desire to contribute to the development of the service. The data collection was carried out over a period of three months.

2.6.2 Make-or-buy decision

Another important method used in this thesis is the make-or-buy decision. After the focus groups interviews were completed, the data was refined in the form of requirement specification. With the specification in place, evaluation of COTS products was done. The evaluation resulted in listing top three products that fulfilled most requirements. Those three products were then compared against potential in-house developed solution. This comparison was done as make-or-buy decision.

Make-or-buy decision is used to evaluate the costs and benefits of purchasing a solution from external supplier, or developing one in-house. The analysis has to include both strategical and operational factors. The strategical factors are often long term and are aligned with already defined company polices, core competences and wanted position for the future.

Example of strategical factors:

- Privacy and confidentiality regulations and requirements
- Intellectual Property Rights (IPR)
- Scalability
- Market position - consumer or supplier of service
- Competitors
- Public perception and other intangible assets

- Integration with existing systems and processes

In addition, operational factors can be:

- One-time costs to build or purchase
- Subscription costs for support and maintenances
- Pricing models
- Scalability
- Possibility to influence the service quality
- Flexibility of the service to be tailored or integrated

It is important to note that while operational factors are likely to be quantifiable and measurable in terms of cost, in contrast, some strategical factors have non-monetary value, which makes comparison difficult. The methods and terminology used for the make-or-buy decision are based on number of studies [4], [6], [76]. However, this thesis does not user the "Theory of Constraints". Instead, own metrics were defined in order to better represent the costs involved and help the decision making process.

This concludes the Literature review chapter of the present thesis. The following Analysis chapter discusses technical details and evaluates the information collected during the focus group interviews. As a result, it will produce requirement specification documentation and present proposal for the make-or-buy decision.

3 Analysis

The previous Literature review chapter covered the basics of software security, how it impacts our daily life and the methods that will be utilized in this analysis.

This chapter will start with introduction to vulnerability management and expand on how it is applicable in the product life cycle. Then, it will present the ISO standards which are commonly used in the industry to identify and handle vulnerabilities. The chapter continues with description of VMS, and the process of collecting requirements for this new service. At the end of the chapter is the Make-or-Buy analysis.

3.1 Vulnerability management

Vulnerability management is becoming increasingly important in our connected society. Furthermore, vulnerability handling has become crucial for IT companies, as security breaches can be fatal for them. Vulnerability management can be seen as a subset of much larger risk management activities. Forman [16] suggest that vulnerabilities are found in environment, economics, technology and strategy.

In this thesis, focus is placed on the vulnerabilities in software and products that use software. In many cases vulnerabilities found in other areas are handled in a completely different manner, for example with contractual obligations. Furthermore, different types of vulnerabilities will be handled by completely different units in large organizations. This often leads to creation of specialized policies and processes, which are applicable only for specific domain. A policy, which covers all aspects of a large corporation, will often be too vague and abstract to be useful for practical guidance. This paper, on the other hand, aims to provide actionable data and lead to the deployment of a VMS for product life cycle.

Forman [16] also highlights a problem in the security industry, related to the false understanding that a product can be developed and security can be added later on. This way of thinking creates a market for solutions which minimize the exposure of vulnerable products, or treat only the symptoms. However, such approach does not address the fundamental problem that security should be addressed throughout the product life cycle.

To be able to define strategy, policy, process and implementation, it is important to have common understanding of the terminology in the field. It is popular practice in the industry to use standards to enable interoperability between companies and establish common language. In this thesis there is a number of applicable ISO standards, which define terminology and processes for communication and handling of vulnerabilities.

3.1.1 ISO standards

It is common knowledge that large companies excel in processes. Using well established standards as ISO helps companies design and deploy scalable processes and enables them to integrate with existing processes and partners. In vulnerability management context the most relevant standards are: ISO/IEC 29147 Information

technology - Security techniques - Vulnerability disclosure and ISO/IEC 30111 Information technology - Security techniques - Vulnerability handling processes.

In addition, it is expected that the company will have implemented the ISO/IEC 27000 family of standards, thus having Information Security Management System in place. Standards are also important when defining terminology. Having standard based vocabulary enables clear and precise communication with suppliers, customers and other partners. To prevent loss of information in this work, terminology from the standards is directly quoted.

ISO/IEC 27000 defines management system as a "set of interrelated or interacting elements of an organization to establish policies and objectives and processes to achieve those objectives". It also defines vulnerability as a "weakness of an asset or control that can be exploited by one or more threats" [26]. In practice, this means that to have vulnerability management system, an organization must have policies and objectives defining how weaknesses in their assets or controls must be mitigated to avoid threat exploitation. Those policies are then implemented in processes to achieve the set objectives.

ISO/IEC 29147 enables companies to: 1) define standard vulnerability disclosure policy which ensures that found vulnerabilities are being addressed; 2) minimizes the risk from those weaknesses; 3) supplies sufficient information to users, which in turn enables them to evaluate the risk to their system; 4) promotes communication between all parties involved [25]. While the disclosure policy is important tool for communication, it is not the focus of this thesis. However, the related ISO/IEC 30111, which defines vulnerability handling process, is a valuable guidance for establishing the policies and processes needed to create VMS, which is the topic of this paper.

ISO/IEC 30111 describes the relation between ISO/IEC 29147 and itself. ISO/IEC 29147 serves as an input to the vulnerability handling process. Then communication originating from the vulnerability analysis is propagated to customers and partners via the channels established as part of the vulnerability disclosure process. The first step in ISO/IEC 29147 is the development of vulnerability disclosure policy, followed by development of capabilities to receive and disseminate vulnerability information.

ISO/IEC 30111 starts with the development of vulnerability handling policy and organizational framework. Once those steps are completed, the company should be able to receive vulnerability reports from external sources, or to proactively monitor for such. Alternatively, the company can identify vulnerabilities from internal sources.

When new vulnerability is received, the company should acknowledge the receipt and verify the report internally as part of Vulnerability handling process. As soon as the vulnerability has been analyzed, the finder should be informed if the company was able to reproduce it or not (ISO/IEC 29147). If the vulnerability is confirmed, the company shall proceed and develop solution to mitigate it. The solution is then verified to ensure that it mitigates the vulnerability and information is sent to partners in the form of advisories (ISO/IEC 29147). In addition, internal post-resolution activities are started. [24]

3.1.2 Definitions

To ensure that vulnerability handling service is compatible with existing tools and enable automated processing of vulnerabilities, it is necessary to use standard definitions, terminology and metrics. Using the ISO and NIST [43] standards as basis, this paper uses the following definitions:

- Vulnerability - is a flaw or weakness in a product which can be accidentally triggered or intentionally exploited to compromise network confidentiality, integrity and/or availability of customer's asset.
- Vulnerable - the product is using the vulnerable component and there may be potential attack vectors.
- Affected - the product includes the vulnerable software or component.
- Severity - the criticality of the vulnerability, related to the impact if the vulnerability is exploited.
- Priority - expression of the time constrains in which a vulnerability has to be handled.

There is number of sources and ways to define vulnerability severity. It can be defined by the security researcher who discovered it, by the software vendor who supplies the vulnerable 3PP, or by the integrator, which builds a product with a vulnerable 3PP. The severity of vulnerability often depends on the exposure of the vulnerable component. This concept is represented in CVSS version 3 scoring [12], temporal and environmental metrics. To be able to decouple the severity of a reported vulnerability from the processing time constrains, a company can decide to implement priority labels. This allows an enterprise to evaluate a particular vulnerability as more important than what is suggested by the public severity score. By assigning higher internal priority label, vulnerability processing is forced into shorter time-to-fix track.

Furthermore, throughout this paper the following industry standards and frameworks have been used:

- Vulnerable components are identified with the help of Common Platform Enumeration (CPE) [35].
- Classification of vulnerabilities in this paper is based on Mitre's Common Weakness Enumeration (CWE) [34].
- Common Vulnerabilities and Exposures (CVE), are used for reference to published vulnerabilities [36].
- The scoring of vulnerabilities is based on Common Vulnerability Scoring System (CVSS) version 3 [12].

- Communication of vulnerability information is handled by ICASI's Common Vulnerability Reporting Framework (CVRF) [23] and its evolution OASIS's Common Security Advisory Framework (CSAF) [47].
- The use of these standards and frameworks enables the Security Content Automation Protocol (SCAP), which has major role in implementing efficient, automated vulnerability handling and compliance audit processes [42].
- The assessments of system security and use of SCAP are further improved with the Open Vulnerability and Assessment Language (OVAL) [37].

3.1.3 Process

Once we have established common language, it becomes easier to delve into the vulnerability management process. As described by Gartner [17], vulnerability management process can improve the security in IT environments. They suggest a process, which contains six steps.

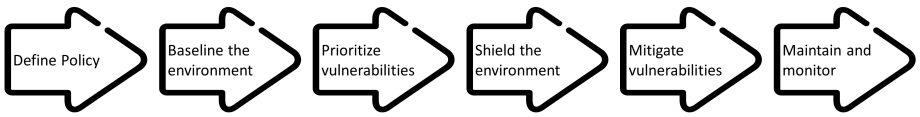

Figure 6: Vulnerability Management Process of IT operations

It is common sense that in order to have effective process to handle vulnerabilities, a structured approach is needed. Regardless if it will be similar to the one suggested by Gartner, or completely different, a company interested in implementing such process should: identify stakeholders; identify assets; collect requirements; and then implement it in a cost effective way. However, the process recommended by Gartner is focusing on IT environments, and would not fit well in IT product life cycle process.

This thesis is focusing on developing VMS for product life cycle. To accomplish this task, first, a common understanding of what is product life cycle is needed. In context of this thesis, the product life cycle is simplified from the ones proposed by Stark [66] and NIST [41], to include the following phases: introduction, growth, maturity and decline. During the introduction, the product is being developed and tested. When the product is released to customers, it enters the growth phase. During the maturity phase the product is being maintained by the company. When the product reaches decline phase, it is eventually decommissioned.

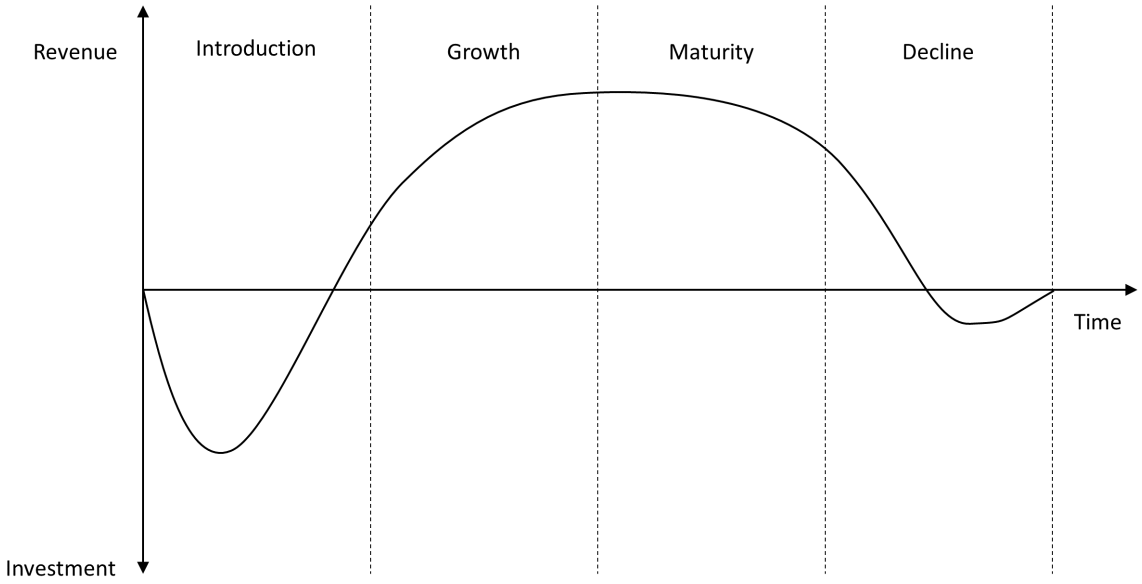

Figure 7: Product life cycle over time

As can be seen from Figure 7, during the introduction phase, the product development requires an investment that is larger than the revenue. The same applies for product testing step. After the product has been introduced to the market, provided it is successful, the revenue becomes larger than the investment and the product starts bringing profit. This characteristic is typical for the the growth phase. The product user base continues to grow until the product reaches maturity phase. At the same time the company is investing mainly in product maintenance efforts. At a certain point, the company can decide that the product has reached its end of life and as the cost to continue maintaining it will grow, it is more efficient to retire it. This happens during the product decline phase.

Later in this chapter, it will be shown that most COTS products for vulnerability management are useful only after the product has been deployed. This will cover mainly the product release and maintenance steps. As most researchers and security companies have low business incentive to publish and detect vulnerabilities for products that are labeled as end of life, most COTS products will not detect vulnerabilities in old or retired products. It is clear that, if a solution should cover the complete product life cycle, it would have a larger set of requirements when compared to others, which handle only a number of phases. In addition, the vulnerability handling process is more complex for companies, which develop IT products, compared to others, which only operate such products.

3.2 Product classification

There are large number of product classifications. They focus on different factors which can characterize products such as: tangible or intangible; hardware or software; consumer or business; bundled or item. In the context of vulnerability management it is important to make distinction between different products based on the possibility

to change them to mitigate found vulnerability. When considering IT products and services, the study will focus on following categories of interest:

- Hardware products
- Hardware dependent products
- Platform dependent products
- Software-as-a-Services products

Hardware products are such where all the functionality is implemented in hardware. Example of such product is a sensor. Changes to the hardware are often expensive and in many case not possible. If a hardware product has to be changed, most often, it is completely replaced by the next generation product. Hardware products are a major challenge for vulnerability management. For low value products, replacement is a viable option for vulnerability mitigation. However, for products with high value and expected life of tens of years, vulnerability management requires workarounds and mitigation by other means.

Hardware dependent products are products, which run firmware or software, but are depended on specific hardware architecture or features. Network appliances could be given as an example of such products. They depend on network accelerator processors and run proprietary operating systems. Nonetheless, it is possible to change their firmware or software to mitigate vulnerabilities. Such products often have long life expectancy and in order to receive correction package, users are required to purchase support contracts. Due to the close integration with hardware, any changes to the software or firmware have to be validated with the hardware as well. That could lead to longer time to fix. In case that support is not provided by the vendor, or product has been retired, it is not possible to receive updates and vulnerability fixes. These products pose medium challenge to vulnerability management.

Platform dependent products are purely software. However, they depend on platforms such as computer architecture or OS. These products allow fixes due to their software nature. If the products have open source code, the customer could implement the vulnerability mitigation directly by changing the code himself. If the products have proprietary code, the software vendor is expected to provide the fix. These products have often short life, usually three to five years. If the proprietary software has reached its end of life, the vendor would not provide fixes. This is a major advantage for open source products, as support and fixes can be done even if the product has been officially retired. Testing and implementing fixes for software products is easier, as they have to be validated only once on the abstraction of platform. Development and testing can be highly automated and fixes can be developed and delivered within hours. Platform dependent products do not pose a challenge for vulnerability management.

Virtualized products are also purely software products. They have been further decoupled from hardware by virtualization abstraction layer. They can still have dependency on OS, however, no dependency on hardware is expected. These products

behave similarly to platform dependent products, therefore they are not distinguished as separate category. Virtualized products do not pose a challenge for vulnerability management.

Software-as-a-Service (SaaS) products are a special type of software products. While the platform dependent products are still delivered to customers, in SaaS case, service is provided for customers. In SaaS scenario, if a vulnerability is found, it is the responsibility of the service provider to mitigate it. Changes happen on layers which are not visible to the service users, thus no actions are required from the users. SaaS allows even more flexibility compared to software products and large service providers as Google, Amazon, Netflix are able to deploy new versions of their service multiple times a day to multiple times a minute [27], [21]. SaaS products do not pose a challenge for vulnerability management.

Having discussed how different type of products relate to vulnerability management, this study continues description of vulnerability management in IT and product development environments.

3.3 Vulnerability management for IT companies

For IT product developing companies the vulnerability management process can include the following steps: Vulnerability monitoring; Patch management; Vulnerability assessment; Software release. Furthermore, it is a cyclic process as we have defined earlier. To improve the understanding of the process, Figure 8 is provided below.

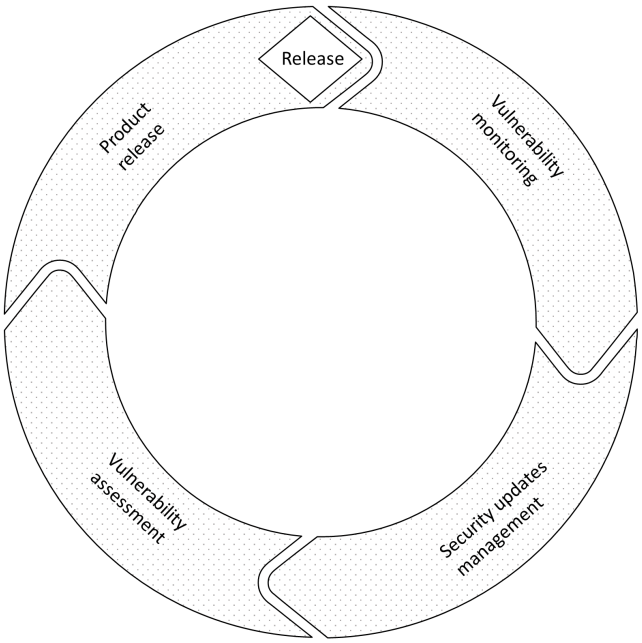

Figure 8: Cyclic vulnerability management process

The first step in the process is identification, analysis and communication of new vulnerabilities. The second step describes how the vulnerability is fixed, upstream components are propagated to product development units and how those changes are tracked. The third step serves as a toll gate to confirm if the fix has been successful and that the product is no longer affected by the vulnerability, which triggered the process. The last step is making the updated software available for download to customers.

In IT VMS, when vulnerability is found, most of the communication is kept internal. It is usually spread in the operations team, or between monitoring and deployment teams if the functions are segregated. On the contrary, in product development, when a vulnerability is found, based on the severity and processes, communication to external parties might be triggered. Thus, it is important to define the communication policy and vulnerability severity which would trigger it.

3.4 Vulnerability management for product life cycle

As discussed above, there are some commonalities in IT operations VMS and own developed product life cycle VMS. In the present subsection, this paper describes the differences and explores further the specific requirements of product life cycle VMS. The inferences made, regarding VMS, are based on the COTS analysis, which is covered later in this chapter.

When VMS has to cover the whole life cycle of a product, it must track each product from its inception, until its retirement. To be efficient, VMS must be integrated with the rest of the product life cycle systems. While IT VMS would be able to provide valuable service for operators in their deployments, it is a bad fit for product development process. Their tooling is focused on detecting vulnerabilities in systems that are operational. IT VMSs are made to detect vulnerabilities on services running on an interface. Those solutions rarely have concepts of product, system or solution.

IT VMS's priority is to execute series of network tests, based on predefined rules, against the system under investigation. Depending on the service response, VMS will suggest if the system is vulnerable or not. This method can lead to false positive results in cases where the detection rule processes the service version, and does not actually try to validate the vulnerability. A vendor can use a version of the software which is considered vulnerable, but without the part of the code which has the vulnerability. This will lead to false positive detection result.

In contrast, product life cycle VMS approach, instead of rules, uses database with listing of all components, hardware and software, commercial (3PP) or FOSS, and internal common components (2PP). It also allows definition of abstract concepts such as system, solution or service, by combining components in hierarchical structure with recursions. This information can be further enriched with customer install base data. Such deployment allows the supplier company to inform its customers when vulnerability is detected in the specific version of the product used in customer deployments, without running scanning tools.

In addition, IT VMS is often providing severity rating based on the assumption that the scanned interface is Internet connected. This often leads to the overestimation of the vulnerability impact, as customers can run the service in restricted networks with additional layers of protection. Furthermore, IT VMS would rarely provide information on such assumption, which can also lead to underestimation of the impact. Simple IT VMS depend on their detection rules, if those rules are not fit for specific customer deployment scenario, the results will be misleading.

In product life cycle VMS, the company which supplies the products has the benefit of knowing how those products are designed and how they are deployed in its customers' premises. This allows product life cycle VMS to provide more accurate rating for impact of detected vulnerabilities. The amount of information that is based on facts is higher when compared to IT VMS, which have to make assumptions instead.

In IT operations when vulnerability is detected by the VMS, analysis is triggered to understand the impact of the vulnerability on the system. The analysis should also lead to proposal for corrective action: software update; configuration change; other or none. In product life cycle VMS, the analysis phase is longer and should produce corrective actions for the products which are already deployed by customer as well as the products that are still under development. Based on the corrective actions proposal, there is wider variety of solutions available when compared to IT VMS.

Having discussed the major difference between IT VMS and product life cycle

VMS, this paper continues with implementation of product life cycle VMS. Based on ISO standards [26] and industry best practices, the establishment of product life cycle VMS requires the following steps:

- Devise vulnerability management strategy
- Write vulnerability management policy based on the strategy
- Define vulnerability management process aligned with the policy
- Implement the process as vulnerability management system
- Deliver vulnerability management service utilizing the system
- Continuously improve the service

The definition of strategy, policy and vulnerability management process is out of scope for this study. Those steps are prerequisites for implementation of product life cycle VMS and should be completed by organization management. In this paper, the analysis of the company's strategy highlighted three priorities in relation to vulnerability management: reliability of information; fast and accurate communication; fast reaction to mitigate vulnerabilities. To summarize, the strategy has three pillars:

- Crowd-sourced vulnerability discovery, utilizing free and commercial vulnerability feeds.
- Scalable service for vulnerability analysis and communication.
- Utilization of existing delivery system to provide customers with fixed software.

Moreover, the policy can be summarized as follows:

- Deliver reliable and relevant vulnerability advisories to customers
- Communicate accurately and in timely manner
- Deliver solution in agreed time frame, based on vulnerability impact.

The next step is the actual vulnerability management process. The purpose of this process is to minimize and control the impact of vulnerabilities on the products throughout the life cycle. The process is cyclic and it applies for a wide range of software products and services, or such that use software. The process applies to different product development methods (waterfall, agile), different software architectures and platforms.

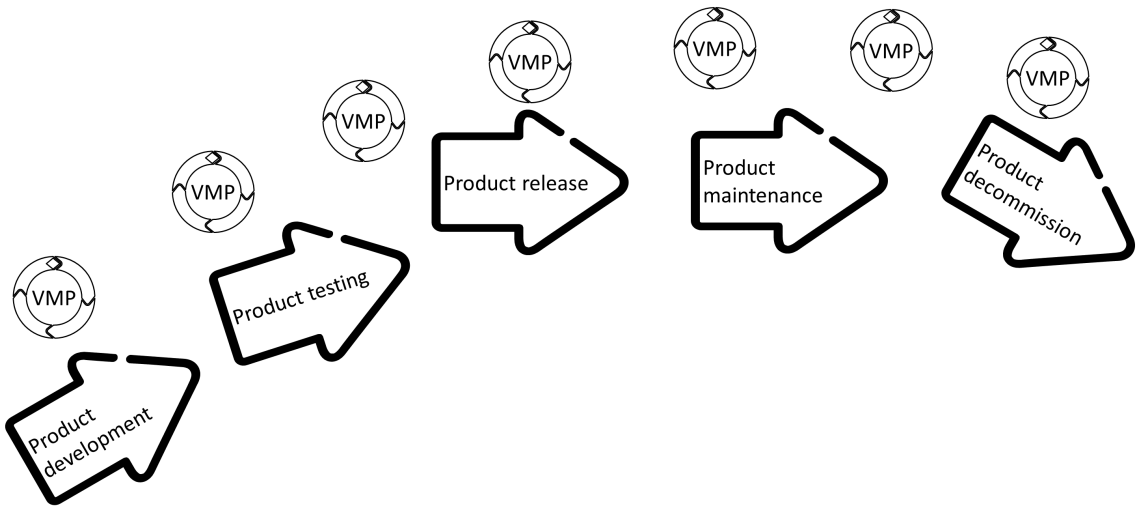

Figure 9: Vulnerability management in product life cycle

Figure 9 depicts how vulnerability management process can be involved during product life cycle. The figure is most relevant for products with software releases. During the product development phase, the process will provide information on component vulnerabilities and serve as a recommendation to developers when they look for a reliable 3PP, which is not prone to vulnerabilities.

During the product testing phase, there is feedback from the test results informing if new vulnerabilities were identified in the product or components. This information is stored and distributed to other users of the same product or components.

During product release phase, the VM process supports security assurance of the product. By combining the test results and the database component analysis, evidence can be provided to customers, confirming that the product is free of known vulnerabilities. In special cases, when it is not feasible to remove all security flaws, there must be supplemental information alerting customers of the potential risk. It is good practice to include also recommended actions to minimize the risk exposure for customer deployment scenarios.

During product maintenance phase, the process ensures that any new vulnerabilities found in the product are communicated in timely manner to customers and developers. Customers can take temporary measures to limit the impact, while developers are working on the patch. When the patch is tested, communication to customers is initiated, providing new software and instructions how to apply the fix.

During product decommission phase, the process is used to communicate to customers the end of the product support and sunset of the vulnerability monitoring activities for the product in question.

Communication is critical for the vulnerability management process. Starting with vulnerability announcement or publication, there should be reliable and trusted channel for communication. The next step is generation of alert to be sent to impacted products, followed by communication to customers and partners. In addition, when impact analysis is completed, the results could be communicated to customers and partners, together with expected date for the fix, or even the fix itself. When

vulnerability information is to be sent to customer, the confidentiality and integrity of the communication channel are of paramount importance. Either interception or tampering with the information would expose the customer further to attacks by malicious actors. Therefore, it is recommended to establish secure channel for communication beforehand. Popular implementations include:

- OpenPGP - IETF - RFC 4880 [7]
- S/MIME - IETF - RFC 5652, 5750, 5751, 5754 [20], [55], [56], [63]
- TLS - IETF - mainly RFC 5246 [10]

Each of the above listed methods have their own merits and drawbacks. In this thesis it is recommend to implement VMS with all of them. This recommendation does not impact the overall cost of the service, while it provides a wide compatibility with human and machine client interface.

3.5 Requirement specification

To design a VMS which caters for the needs of a wide variety of users, in this thesis, the method of focus group interview was utilized, as described in the Literature review chapter. To facilitate productive discussions within the focus groups, the users were divided by role and interest in VMS. The following groups were formed:

- Product security responsible
- Product security adviser
- Product management
- Marketing and sales
- Customer support
- External customer (telecommunication operators)
- Government and regulators (considered, but not implemented as currently there are no legal requirements to implement such functionality).

After series of interviews, the service requirements were aggregated and formalized into a requirement specification document. As most of the users in the focus groups were interested in the service functionality, and not in its implementation, there was a number of requirement areas that were not covered in the interviews. More specifically, the areas identified for additional research were:

- Service availability
- Service and data integrity
- Data confidentiality

- Roles and responsibilities (Segregation of duties)
- Audit trails and logs
- Backup process

The additional requirements in the list above are crucial for implementation of VMS. However, those requirements and their specification are not in the scope of this paper. Regardless, those requirements were specified as part of the commissioned work. Once all requirements have been document and approved by the VMS owners, the work proceeds with analysis of commercial off-the-shelf products which could be utilized as VMS.

3.6 Commercial off-the-shelf products

When selecting commercial and open source products, which could fulfill the requirements for VMS, a number of commercial presentations and whitepapers were reviewed [53], [46], [70], [62], [72], [13], [54], [38], [39], [58], [57], [73], [8]. The once selected for preliminary evaluation are listed below:

- RSA Archer
- Alien Vault USM (Unified Security Management)
- Secunia Vulnerability Intelligence Manager, also know as Flexera Software VIM (Vulnerability Intelligence Manager)
- Qualys VM (Vulnerability Management)
- Tenable VM (Vulnerability Management),
- Tenable SC (SecurityCenter)
- Rapid 7 InsightVM
- Rapid 7 Nexpose
- Veracode VM (Vulnerability Management)
- NCSC-NL Taranis

The initial evaluation included a limited number of features for consideration. It is most important to discover if product is ready to be deployed on enterprise scale with hundreds of active users and to maintain information for hundreds of products. Stability and reliability were considered as subset of enterprise scale requirements. Another important requirement is the communication ability of the system. The summary shows an overview if there is some communication options, and what is their level. Standards coverage is another crucial parameter for evaluation, as it is essential to have system that is interoperable with other existing solutions. It

was also valuable to understand if a product is available as open source code. That could allow modification and tailoring of the system in-house. Another feature based on privacy and confidentiality concerns is whether the system is available as an on-premises solution, or only cloud based. Flexibility shows the configuration options and extensibility of the system. As vulnerabilities are being identified in components (software or hardware), it is critical for the VMS to be able to represent the complete product structure, down to the smallest component that could be vulnerable. Lastly, the cost factor was summarized as price for solution, which will accommodate the estimated workload. A quick summary of these features is provided in Table 2 below.

Product	Enterprise scale	Communication & Reporting	Standards coverage	Open Source	On premises	Flexibility	Price
RSA Archer	Yes	Basic	Basic	No	Yes	Advanced	High
AlienVault USM	Yes	Basic	Poor	No	Yes	Basic	Low
Flexera VIM	Yes	Advanced	Advanced	No	Yes	Advanced	Very high
Qualys VM	Yes	Basic	Basic	No	Yes	Basic	High
Tenable VM	Yes	Basic	Basic	No	No	Basic	High
Tenable SC	Yes	Basic	Basic	No	Yes	Basic	Medium
Rapid7 insightVM	Yes	Basic	Basic	No	Yes	Basic	Medium
Rapid7 Nexpose	Yes	Basic	Basic	No	Yes	No	Low
Veracode VM	Yes	Basic	Basic	No	Yes	Basic	High
Taranis	Yes	Advanced	Advanced	Yes	Yes	Advanced	Free

Table 2: Features summary

3.7 Requirements specification details

The company has identified the need for a VMS. The scope of the vulnerability management is any flaw that can potentially impact confidentiality, integrity or availability (CIA) of any asset or service monitored. The service should track vulnerabilities, advisories and mitigations. It shall be used to monitor for new vulnerabilities in 3PP and 2PP as well as to record those vulnerabilities. The recorded vulnerabilities could result in alerts sent to product organizations and external customers. The alerts shall be communicated via predefined notification channels. Depending on the severity of the alert and assigned priority label, answers are expected from the product organization (within defined time frame). The answer can be further communicated to external customers.

The service is expected to be secure, to utilize best practices and industry recognized standards. To ensure service security, a dedicated security requirement specification shall be prepared. The service will undergo rigorous vulnerability and penetration testing sessions with each new release. During acceptance testing phase, all security requirements must be verified. The service shall not be put in production unless all security requirements are met.

The service is expected to respond positively to changes in the environment, new trends and technologies. It is expected that the service will adopt new standards and requirements as well as follow trends in security to be able to deliver value to its users.

The VMS should be aware of the functionality, technical architecture and sub-components (including operating systems, databases, servers (HTTP, SSH), middleware and libraries) of the products and solutions being monitored.

3.7.1 Registration

The service shall allow any company product, software, solution or a system to be registered. It shall be possible to specify all components of the registered entity in standard format. As an industry best practice, the service shall support CPE format. All components, which are not found in public CPE dictionaries, shall be stored in custom CPE dictionary. Users shall be allowed to register only components in CPE format. This is done to ensure consistency of the input information and minimize possible duplicate entries due to input errors. The service shall be accepting standard Microsoft Excel spreadsheets as input, as long as the entries inside are in CPE format and are stored according to predefined template.

In addition, the service shall allow security contacts with different privileges and roles to be assigned per product version. It shall be also possible to assign contacts on product level or higher, for monitoring and observation purposes. All registration in the service shall be following role based access. The initial users in the services will be configured manually by the administration team after those users have been vetted. Once a critical mass of users is registered in the system, future user management tasks will be handled by the integrated role base access, permissions and request handling processes.

When new vulnerability is recorded, it shall have a unique identifier (CVE or other). In addition, all recorded vulnerabilities shall have a CVSS version 3 (backward compatibility with version 2) value calculated to analyze the severity of the issue.

3.7.2 Vulnerability monitoring and analysis

VMS shall be used to monitor continuously security advisory sources and to assess those with respect to the assets and systems monitored. The service shall allow manual input of vulnerability information and be able to automatically parse standard feeds (RSS or similar). The service shall have facilities to automatically interpret the majority of vendor security advisories. Many software and OS vendors have web pages to communicate security advisories with their customers. For example, Microsoft, Oracle, Google, Red Hat, SuSE, Ubuntu have standard security advisories

available on their support pages. Internal security tests shall be utilized as input of vulnerability information to complement the external sources.

When potentially vulnerable component is found, CVSS "base" and "temporal" rating shall be calculated. Based on the rating, the process will determine if alert needs to be generated and within what time frame it should be handled. The service shall automatically match potentially affected products based on supplied 3PP list with CPE formated entries. It shall be possible to differentiate between vulnerabilities found in components and ones found in own developed products. It should be possible to assign different impact values based on the source of the vulnerability (component or own software). Also, the recipients of the information are determined based on the CVSS rating. CVSS "environmental" rating shall be calculated when necessary.

Figure 10 presents a summary of the proposed vulnerability analysis process in product development. The VMS shall implement the proposed process, to facilitate structured analysis and aid the software development activities.

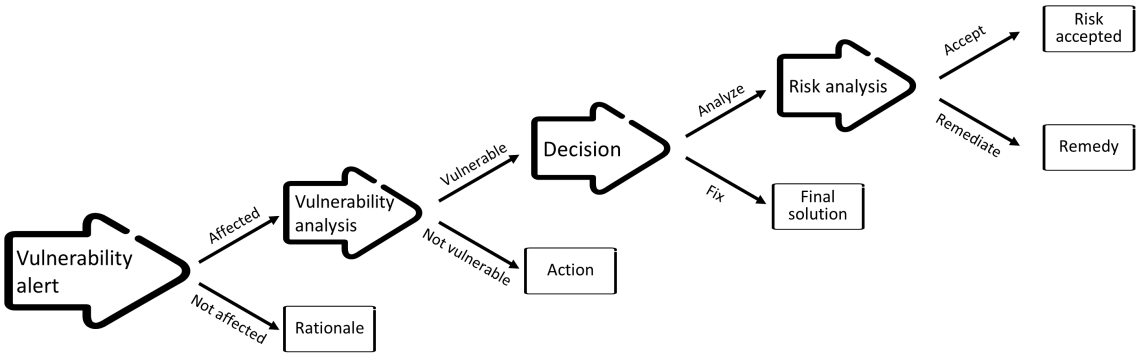

Figure 10: Vulnerability analysis process

3.7.3 Vulnerability remediation

Based on the priority label of the vulnerabilities found, different response time shall be followed. The detailed definitions of severity and expected resolution times are described in the service requirements specification document. The development teams shall verify first if their respective products are affected. Then, analyze the impact of the vulnerability. If change in the product is needed, ticketing or bug tracking systems can be used to trigger change process and ensure tracking of the vulnerability. The security responsible contact shall provide a plan for mitigation and communicate back to the system with their analysis. The response will trigger notification to users with assigned permission to view it and shall be provided in standard form. The formatting and fields in the analysis are enforced by the service. The information input is parsed by the system and rejected if it does not meet the requirements.

After the analysis phase is completed, the service shall be updated if the proposed mitigation or time plan for fix has changed. When solution is available, it shall be tested to confirm that the vulnerability has in fact been mitigated. This could be integrated as part of the regular vulnerability assessments. All vulnerabilities

mitigated in specific software version shall be listed and described in the release notes. This will increase the transparency and build trust with the customers, as they will be able to see and verify that fixes were implemented as agreed.

3.7.4 Communication methods

Any communication regarding vulnerabilities (even not confirmed ones) shall be done via encrypted channels. Vulnerability information is classified and company confidential and thus can be transferred only in encrypted form. The service requirements specification has detailed information regarding the encryption methods, cipher suites and algorithms that can be utilized in different environments (web UI, e-mail). When necessary, additional communication could be used. For example, in case of critical vulnerability, it would be important to communicate to customers using different channels, enabling wider reach. All vulnerabilities which are known to affect the product should be included as part of the product documentation. The documentation should provide description of the vulnerabilities, status of their mitigation, severity level and impact. Security advisories which are applicable to the product version could also be included as part of the product documentation.

In addition, the service shall allow customer specific vulnerability notification and advisories, with approval steps and translation when needed for specific markets. The service shall provide notification and advisories in human readable format and machine readable format simultaneously. This allows customers to integrate it with their internal systems and automate the vulnerability processing.

All vulnerability notifications and advisories shall have permanent addresses. This allows customer to reference the information, relying that it will always be accessible on the same URI.

The service shall provide customers with interface where they can generate and view reports on vulnerabilities. At minimum, the service shall allow per product (vulnerabilities) reports, per vulnerability (affected products) reports and status reports. In addition, the service could provide solution follow-up reports. The service shall allow customers to schedule regular/recurring reports which will be generated automatically. The service shall use CVRF as a default communication framework.

The service shall enable systematic handling of vulnerabilities. This will allow the inclusion of a list of mitigated CVE (or other ID) as a part of the software release documentation, as already mentioned. In addition, it can be used to provide status of vulnerabilities that are not yet fixed but have mitigations also as part of the release documentation.

As VMS is taking advantage of available vulnerability feeds, it can also be used to bring value to the security research community. When vulnerabilities are found as a part of the regular VA activities in own products or 3PP components, VMS can be used to facilitate responsible disclosure and communication with suppliers and regulators.

3.7.5 Service and Technologies

The service shall support integration with existing company systems and processes. It shall be able to automatically import product information in standard format for data warehouse or similar system. The service shall be able to integrate with existing directory services, which are used as a source of contact information and personnel changes. Existing software release systems should be allowed to trigger changes in the service and notify users when action is needed.

The service shall also allow integration with export regulation and license compliance systems. When new software component is on-boarded, it should trigger changes and add the new entry for vulnerability monitoring. The reverse connection shall be allowed as well: when a severe vulnerability is detected in a particular component, VMS shall trigger changes in software repository systems and ban the component from further use, until the vulnerability is mitigated. If the mitigation involves upgrading to higher version, VMS can trigger sourcing process too. All automatically inserted data shall be verified by a person with privileges to commit the change to the service database.

The information collected in the VMS is confidential and it has a high value for malicious actors. For these reasons, VMS has strict security requirements. Starting with physical security, the system shall be placed in "Red" zone, where only limited staff has access. All physical system access shall require two factor authentication process. Biometric sensors are optional. The system shall be constantly under video surveillance. Any maintenance and facility personnel shall be escorted by security guard when performing their daily tasks.

Furthermore, the service shall be protected by implementing defense in depth principles. There shall be high availability physical firewall appliances to filter network traffic and prevent abuse. VMS shall be managed by a limited number of administrators and their access shall be logged on dedicated remote server to prevent tampering with the evidence. Two factor authentication shall be required for all management tasks. All data shall be encrypted, both in rest and in transit. All system keys shall be stored is specialized Hardware Security Modules (HSM). This will prevent theft of the keys and minimize the risk of malicious actors being able to spoof the identity of the system. HSM are also able to protect the encryption keys used for communication and data at rest protection.

All communication originating from the service shall be digitally signed, to ensure integrity protection of the data. In addition, logs and audit trail shall be signed and encrypted when transported to the remote server. All system access actions shall be traceable to individual person. No system or group accounts are allowed for service management tasks. If desired, ISMS can be implemented and the service can be ISO 27000 certified. The service shall also support cryptography based time stamping to prevent replay attacks and tampering with the data.

After a thorough review of VMS available on the market, three were selected for final evaluation in the Make-or-Buy decision section:

- RSA Archer

- Flexera Software VIM
- NCSC-NL Taranis

3.8 Make-or-buy decision

The make-or-buy decision is critical point in this study. To aid the decision making process the factors in the analysis were classified as strategic or operational. Table 3 shows a summary the factors in each category.

Strategical factors	Operational factors
• Privacy and confidentiality regulations and requirements	• One-time cost to build or purchase (also known as Capital Expenditure - CAPEX)
• Integration with existing systems and processes	• Support and maintenances cost (also know as Operating Expenditure - OPEX)
• Scalability	• Flexibility of the service to be tailored or integrated
• Need to share confidential information with third parties	• Attack surface of the service
• Trust in the service providers	• Competence needed to use and support the service
• Benefits of internal synergies	
• Influence on brand and reputations	

Table 3: Top strategic and operational factors

The final three COTS product were evaluated based on identified strategical and operational factors. Table 4 provides overview of the benefits and drawback for buying VMS.

Benefits	Drawbacks
• Quick deployment, service offering with possible on-premises appliance • Industry standard tools that have been tested and used by many, reliable source of information • Lower CAPEX	• Pricing scales with number of products and number of users. High OPEX • COTS solutions able to cover only parts of the product life cycle • Additional tools needed to be developed in-house to fulfill all requirements not covered by the COTS product • Inability to represent complex product structure • Potential risk of exposure of confidential information • Inability to re-sell the service to third parties • Open source product does not provide support, thus the company has to invest and create own competence and support capabilities

Table 4: Benefits and drawbacks of buying VMS

As the company had strategic interest in security services, more factors were identified in favor of developing own VMS. Table 5 depicts the analysis of benefits and drawbacks for implementing own VMS.

Benefits	Drawbacks
• Ultimate confidentiality, no information leaves the company and can be controlled to comply with existing company policies • Integration with existing tools and processes • Medium OPEX which grows only when more resources needed • Scalable solution to accommodate the complete company portfolio without additional cost • Flexible data models and representation allowing custom product structure and hierarchical relations • Possibility to sell services based on own VMS • Intangible benefits to reputation as company capable to develop owns solution	• Longer time to deployment • Risk of flaws and lack of experience • High CAPEX

Table 5: Benefits and drawbacks of making VMS

Even though the factors listed above were important for the decision making process, only by implementing own VMS the company will be able to fulfill all requirements that were collected. Having considered the potential cost of development and operating own VMS, the author recommends to develop own VMS as more cost effective solution. The company management weighed all the factors and accepted the recommendation to implement own solution. Developing own VMS was also considered needed as part of internal vulnerability management improvement program. The following chapter contains the design and implementation steps taken to deploy own VMS.

4 Solution

This chapter presents a summary of the actions taken to develop and deploy own VMS.

To be able to utilize efficiently such service, there is a need for organization wide process and culture changes. This starts by devising vulnerability management strategy on company level. The strategy should embody the vision of the company for vulnerability-free products and services. Once the strategy is in place, the next step is to define policies, roles and responsibilities. These steps are performed by high and middle level management. The next steps require more field specific knowledge and expertise, thus it is recommended to be performed by vulnerability management and security experts.

The experts should define the actual processes that will be used in the company and the interfaces of those processes. They should also specify how the new service will interact with the existing systems and define interfaces and APIs for future extensibility. When the details have been documented and approved by the management, the project can proceed with even more technical requirements. At this stage, it is advisable to involve service design, user experience (UX), network and security experts. At this point of the project. it is recommended to prepare a resource plan. An example of such plan can be seen on the Figure 11 below.

Role\Timeline	January	February	March	April	May	June	July	August	September	October	November	December
Customer												
PDU												
PDU												
Tester												
System Administrator												
Developer												
Developer												
VM expert												

Figure 11: VMS development resource time plan

Based on the plan and the identified resource needs, company management should commit and secure funding for the VMS development. When developing service of such importance, it is recommended to have full-time dedicated personnel working on the project. Temporary assignments and personnel changes add unnecessary strain on the development process. In addition, it is very valuable to assign full-time project manager or lead person, which will make sure that the service is being developed as required and any impediments are promptly addressed. The project lead should also communicate to the stakeholders progress reports as well as any additional needs identified.

4.1 Service design

There are different representations of service architectures. Value network configuration is one way to depict multiple actors co-creating value as a part of the services. The proposed VMS is rather complex, thus, for simplicity reasons, some of the technical

and business interfaces, which are not a focus of this study, are omitted. Figure 12 depicts the simplified configuration of VMS.

The core of the service is vulnerability database, which stores vulnerabilities and their relation products and components. There are a number of technical interfaces utilizing service APIs. They allow communication and collaboration within the company, as well as with external partners. For example, 3PP vendors can provide vulnerability feeds to VMS with RSS, or a direct subscription. Once a component is selected for use in the company, it will be stored in the central repository. The repository can communicate with VMS database to register new components for vulnerability monitoring. VMS can, in return, update the repository with information regarding vulnerable components, which should not be allowed for use in products. PLCM systems can communicate toward VMS database information on product life cycle state. When a product is retired, there is no need for vulnerability monitoring process. In addition, when a new product is released, PLCM can communicate and register the product and its components in VMS.

When a new vulnerability is published, the information is recored in VMS. Then, alert notifications are sent to product development units and customers. Once product units have evaluated the impact, the analysis is recored in VMS and forwarded to customers. After mitigation measures are in place, provided new software is released, product delivery organization will update VMS accordingly. At this point, product release directly or via VMS can inform customers that new software is available for download, which mitigates the vulnerability in question.

The are two special cases in VMS: communication with security researchers and with regulators. While communication with regulators is based on legal requirements and partnership agreements, there is no need for a specific technical interface. It is expected to be a rare case, and could be considered in future studies. On the contrary, communication with security researches will benefit from structured technical interface. The interface in question should be specified as a part of the vulnerability disclosure policy of the company. It could be implemented as a simple e-mail box or, it can even have a dedicated web interface, if rich functionality is expected.

This concludes the service overview. The next step in this study provides technical details on the service design.

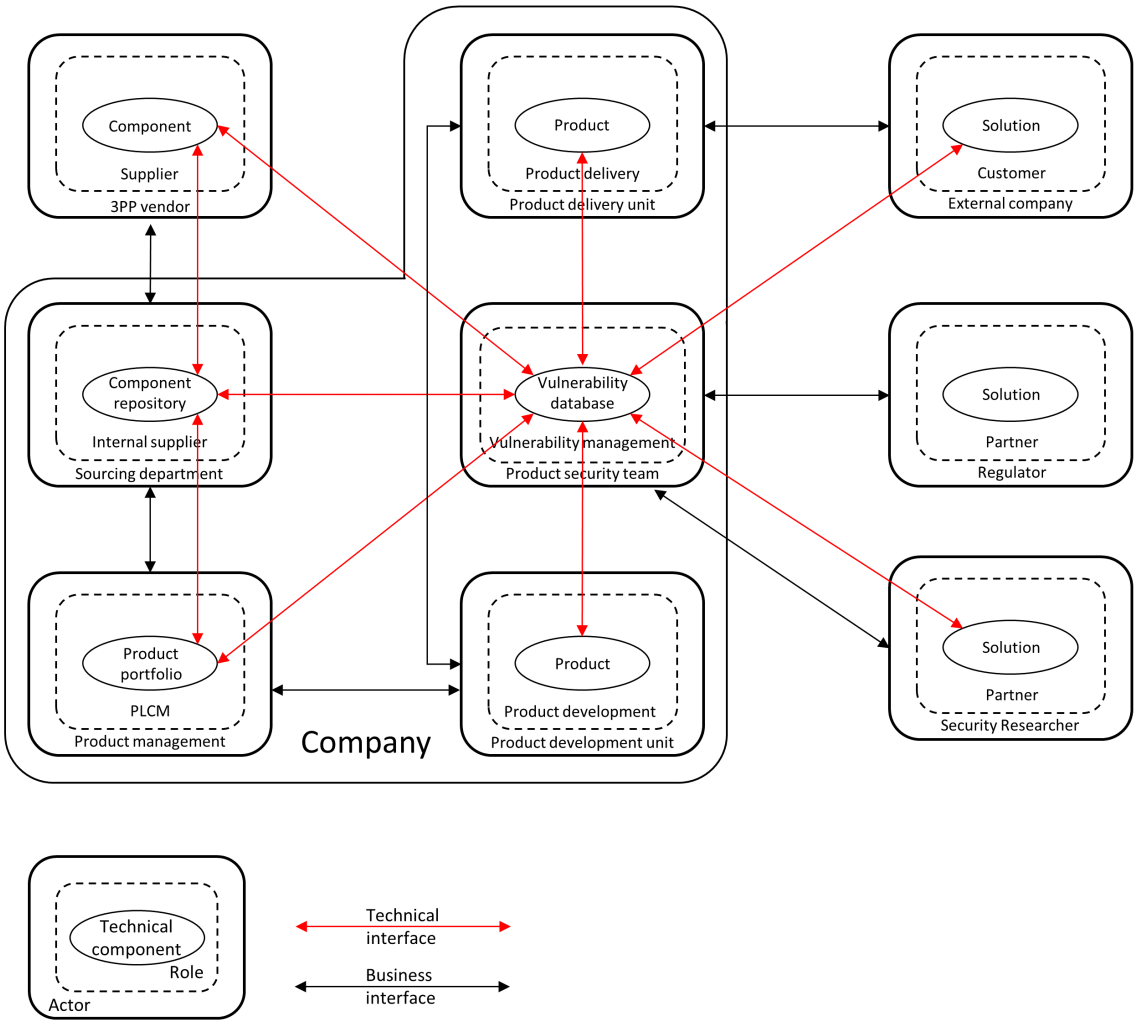

Figure 12: VMS value network configuration

Since VMS is a software service, industry best practices suggest the need for development environment, test environment, verification environment and when the service is ready, production environment. The service will have to be developed over a long period of time and further improved after being deployed. It is advised to have well established development and change management processes in place. All applicable company IT and security polices and requirements shall be implemented as well.

It is also beneficial to utilize again the focus groups, which were formed to collect user requirements. Another option is to look for pilot users and develop a prototype in close collaboration with such test users. Short feedback loop from the users and rapid release software cycle are also beneficial for service development. After reviewing the collected and approved requirements, the development team should make service architecture and design proposals for review. Having Proof-of-Concept (POC) demo is another valuable tool. Mock-up designs can be utilized to gather input from the selected users, thus having better specification of the UI, before time is spent on actually developing it.

In parallel, the vulnerability management experts should work together with the developers to specify the data models and concepts that will be used within the VMS. As soon as the data model and structure have been approved, privacy impact analysis should be performed. Based on the results, refinements to the data models and structure can be made to decrease the impact if necessary. If the refinement steps are insufficient, additional controls should be specified and implemented. Privacy violations have serious financial consequences for the company, which commits them. Thus, there is a strong incentive to mitigate as many as possible, and minimize the risk of such violations.

At this point, the VM processes should be translated to service work flows. With the parallel tasks for data modeling and POC, UI mock-ups also in progress, the project lead should specify the duration of development and review cycles. As already mentioned, having short feedback loop with users is very valuable. However, having too short development sprints is not always efficient. A balance should be reached between short feedback loop and optimal development cycles.

Following the latest software design principles, the service implementation should be broken into small independent components, which can be designed and tested separately. Then continuous integration and continuous deployment machinery will be responsible for testing and deploying the newly built components. Regardless of the actual methods used for service development, the project should have a clear acceptance testing procedure and definition of done. Industry best practice recommends to use a checklist, based on the service requirement specification for such purposes, which is signed-off by the stakeholders.

The project lead should have a release plan and a road map, showing when specific requirements or service features will be implemented. The development team should agree with the stakeholders on a deadline for the minimal viable delivery, when the service is deployed in production setup and pilot users are engaged in testing the implementation. When the deadlines are set, activities, which accompany the release, should be specified as well. These include risk assessments, vulnerability assessments, penetration testing and re-run of the privacy impact assessment. Additional personnel may be involved as experts to execute the assessments. While the development activities are ongoing, the same team, or a separate one, should also consider a number of service implementation requirements.

4.2 Service implementation

VMS will hold confidential information with high value for attackers. Based on the results from the risk assessment, risk treatment plan shall be made and implemented. Any residual risk, which is not mitigated, shall be signed-off by the stakeholders. Best practices and industry standards shall be followed. To highlight a few:

- No clear text communication shall be allowed
- Role based access control (RBAC) and host based access control (HBAC) shall be implemented

- Segregation of duties shall be in place
- Administrative actions shall require multiple factor authentication mechanisms
- Administrative actions may require M of N principles, where from N-number of account holders, a minimum of M-number are needed simultaneously to perform an action. This practice is common for key creation ceremonies or audit events.

In addition, when building VMS, which will be used by the whole company, Business Continuity Management (BCM) must be implemented. Service Level Agreements (SLA) must be defined and approved by the stakeholders. Consequences of violated SLA shall be specified as well. BCM is often supplemented by Disaster Recovery Planning (DRP) activities. Depending on the SLA and requirements, service design should consider Geo-redundant solution. Off-site stand-by system is an alternative way to fulfill high-availability requirements.

As discussed earlier, ISMS is an important part of secure system design. In addition, if the company has decided that security certification is needed, ISMS must be implemented to fulfill ISO/IEC 27000 requirements. Risk assessment, vulnerability assessment and penetration testing are a part of industry best practices too. VMS is expected to have long life, thus it is necessary to define its own life cycle management processes, including operations, maintenance and further development.

If it is deemed necessary by the stakeholders, security benchmarks can be executed as well. This will provide additional assurance for the posture and implementation of the services and can be reused later, when offering the service to third parties. An example of such benchmarks are the Center for Internet Security (CIS) security benchmarks [15]. Another industry accepted reference is Open Web Application Security Project (OWASP) [48]. While CIS benchmarks are focused on measuring compliance, OWASP projects include tools, recommendations, or just lists of risks [49].

Having discussed some of the technical details about service design and implementation, it is important to describe the processes that will be implemented by VMS. The VMS processes design and implementation are described in the following subsection.

4.3 Vulnerability management processes and maturity model

Figure 13 below shows a simplified overview of the way VMS is interacting with the other processes and systems in the company. It is often costly to implement the complete service and deploy it only when all of its components are in place. A more practical approach is to define a Minimum Viable Product (MVP), which will satisfy just enough of the user needs and provide valuable feedback for further development. This approach allows the segments of the Figure 13 to be implemented and deployed gradually. Design and deployment with MVP approach minimizes the risk and upfront investment needs.

Furthermore, it is possible to integrate the releases of different segments of the service with the vulnerability management maturity model on company level. By

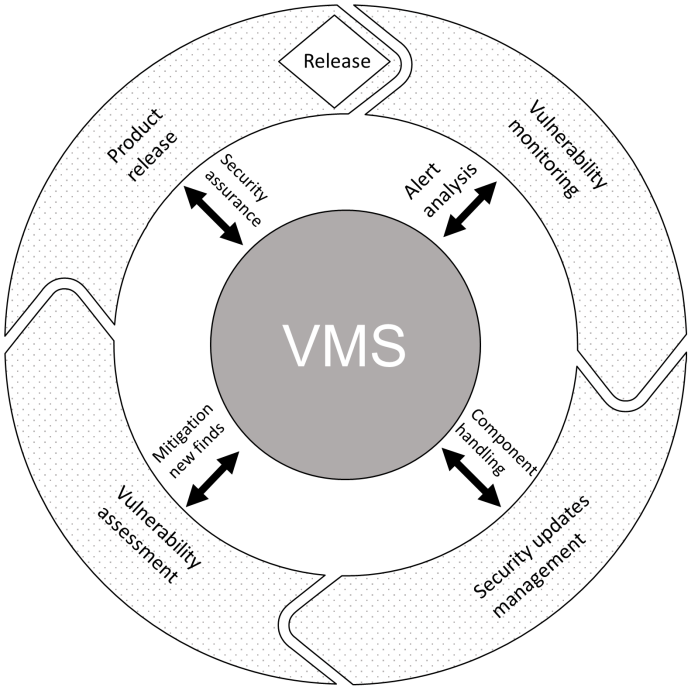

Figure 13: VMS process

deploying a new service segment, the vulnerability management would advance to the next maturity level. This also allows the service deployment progress to be easily visualized and communicated to high level management. If the company has ambition to reach a certain level of maturity, only what is needed to reach that level will be developed and deployed. This close relationship with the business needs allows for efficient and cost effective service design and deployment process.

The following subsections present details on the processes and information flows required for effective enterprise VMS.

4.3.1 Vulnerability monitoring

The first depicted step in Figure 13 is vulnerability monitoring. The input for that process comes from a number of vulnerability sources. Free and commercial feeds can be utilized for this purpose. The Product Security Incident Response Team (PSIRT) or similar Product Security Team (PST) will analyze the vulnerability information and record it in VMS. VMS in turn will match the vulnerable components to company products and send notification to the users with assigned privileges.

These vulnerability alert are processed as part of the product development and maintenance activities. However, no more information is provided to VMS on

implementation of fixes, verification or delivery time frame. The vulnerability monitoring process is shown on Figure 14 below.

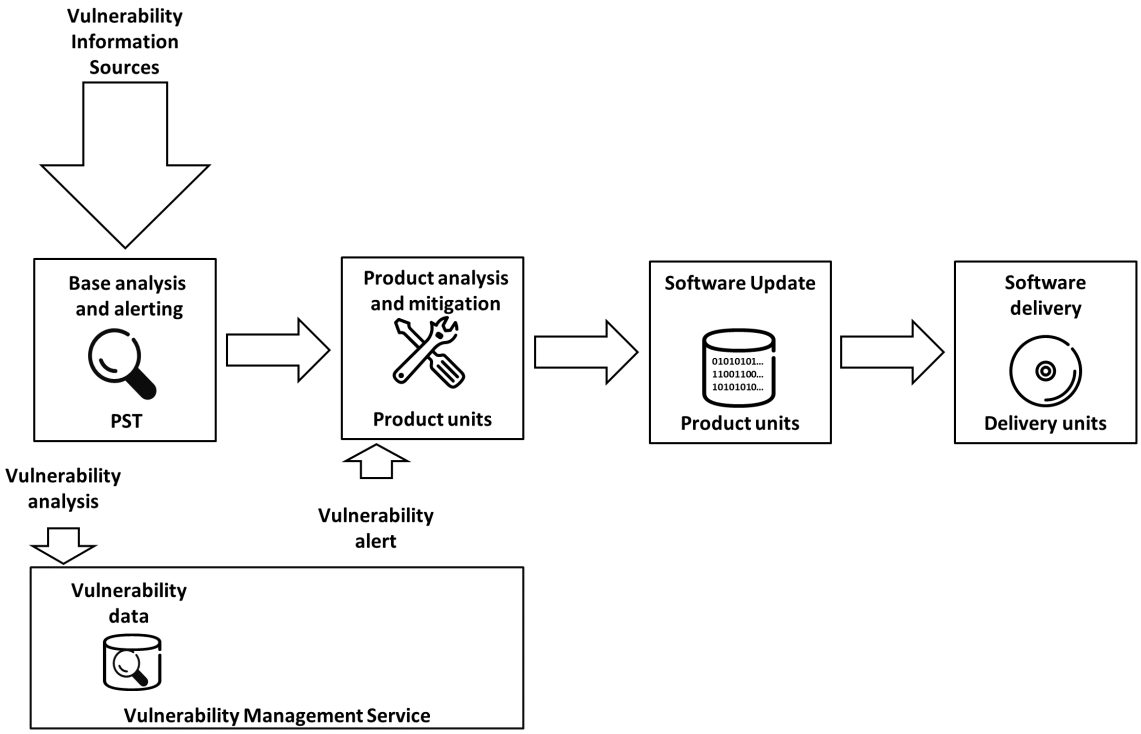

Figure 14: Vulnerability monitoring process

4.3.2 Security updates management

The next step is security updates management, shown in Figure 13. As part of the security updates management process, product development and maintenance are expected to provide information on any mitigation steps that are planned. These organization should also provide information on potential software releases, in which the vulnerability will be mitigated. Large software development companies would often utilize dedicated design organization for software bug corrections and general software updates. A separate organization can be used for security updates development and release.

In most cases, security updates to software packages are bundled with feature updates. Mitigating a vulnerability often requires the uplift of the component versions. Such version changes trigger the need for sourcing organization to review and approve the new version for use. In addition, VMS can add labels to vulnerable component versions in internal software repositories and prevent further use of vulnerable components in product development. The addition of security updates management process is shown on Figure 15 below.

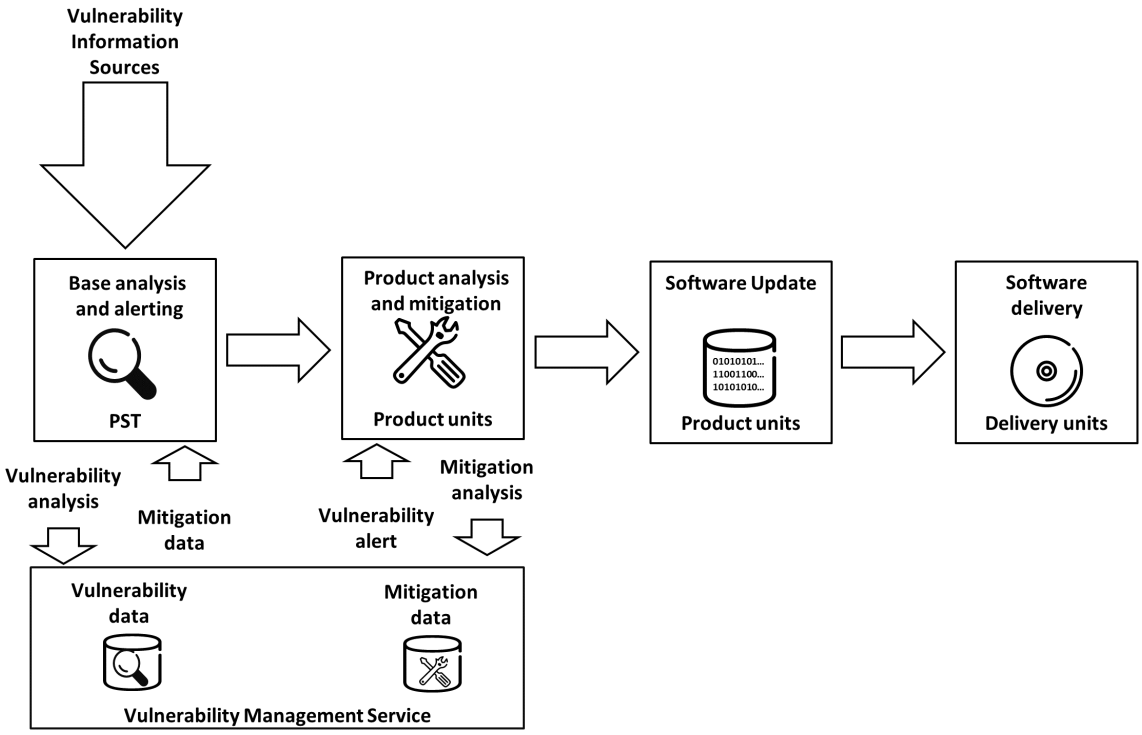

Figure 15: VMS with added security updates management process

4.3.3 Vulnerability assessment

The third step on Figure 13 is vulnerability assessment. Vulnerability assessments are often performed in product development and maintenance as a part of their regular testing process, or as an additional security testing process. In VMS context, VAs must be used to verify that all vulnerabilities, for which the product unit has received alerts, are mitigated. In addition, VAs can be utilized as sources for new vulnerability information. Fuzzing tests and manual exploit testing are examples of methods which allow detection of previously unknown vulnerabilities in a product. To increase the value of the assessments, it is critical to automate the process as much as feasible. This will allow organizations following Agile methodology, or having frequent software releases, to align their development process with the vulnerability assessments.

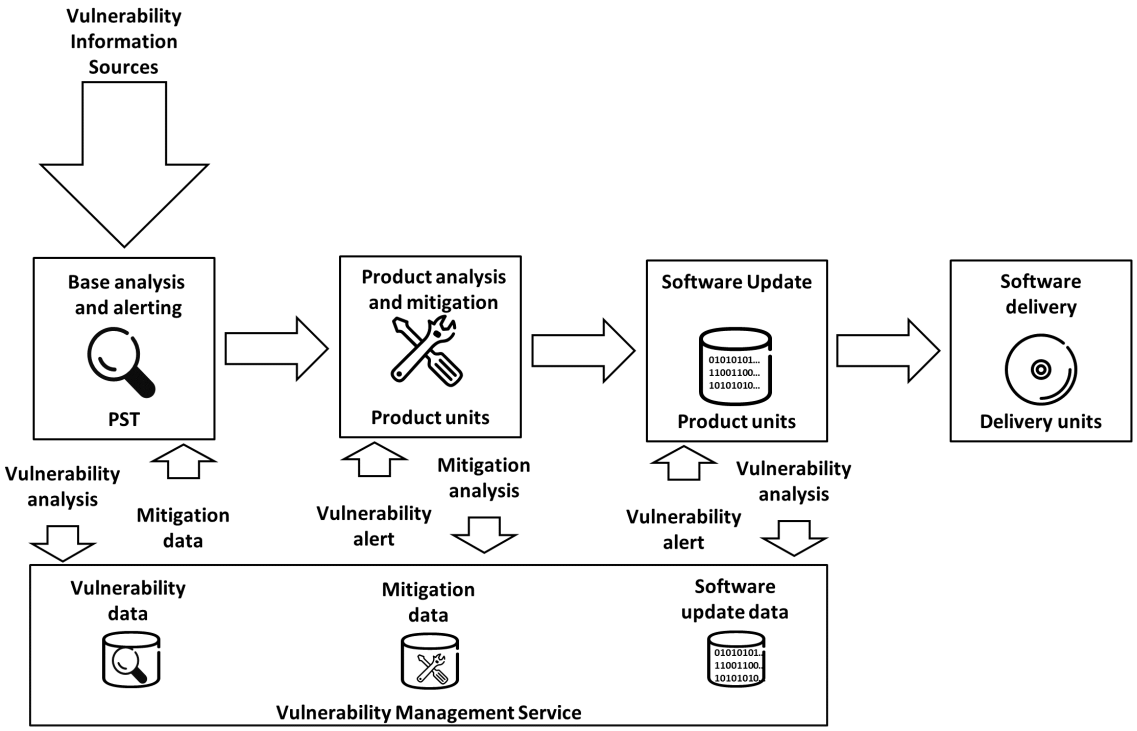

Figure 16: VMS with added vulnerability assessment process

As discussed earlier, VMS can be used as an input for the sourcing process, as seen on Figure 16 above. Vulnerability alerts are fed to software update process in sourcing and product development units. This is another activity, which must be automated in order to maximize the value delivered by the service. When new, fixed software is available, product development and maintenance units shall run vulnerability assessments to verify if the vulnerability is mitigated. The same process shall be followed regardless of the source of software updates. External and internal software fixes shall be tested to provide evidence of successful vulnerability mitigation.

Due to governmental and contractual requirements many software developing companies would run standard vulnerability scanners against the ready for release product. This is done to provide assurance that all known vulnerabilities have been identified and addressed. The scans also assist in the detection of possible false positive results and possible misinterpretations of such scanner results, when run by customers.

4.3.4 Product release

The forth step on Figure 13 is product release. When the vulnerability alert has been processed, the software has been updated and evidence have been provided that the vulnerability in question has been mitigated, the product can be released to customers. At this step, vulnerability alerts serve as input for the product documentation, such as release notes. Product release notes shall contain references to all vulnerabilities that have been mitigated as a part of the product development and maintenance process. It is recommended to use standard reference for each vulnerability such

as CVE. Product release can also provide input for VMS in the form of security assurance. VMS database shall be updated to reflect that new product version has been released and shall be monitored for vulnerabilities. In addition, VMS database shall be updated to record the version and release date of the software, which has mitigated vulnerabilities. The data should also reflect the availability of the new software to customers.

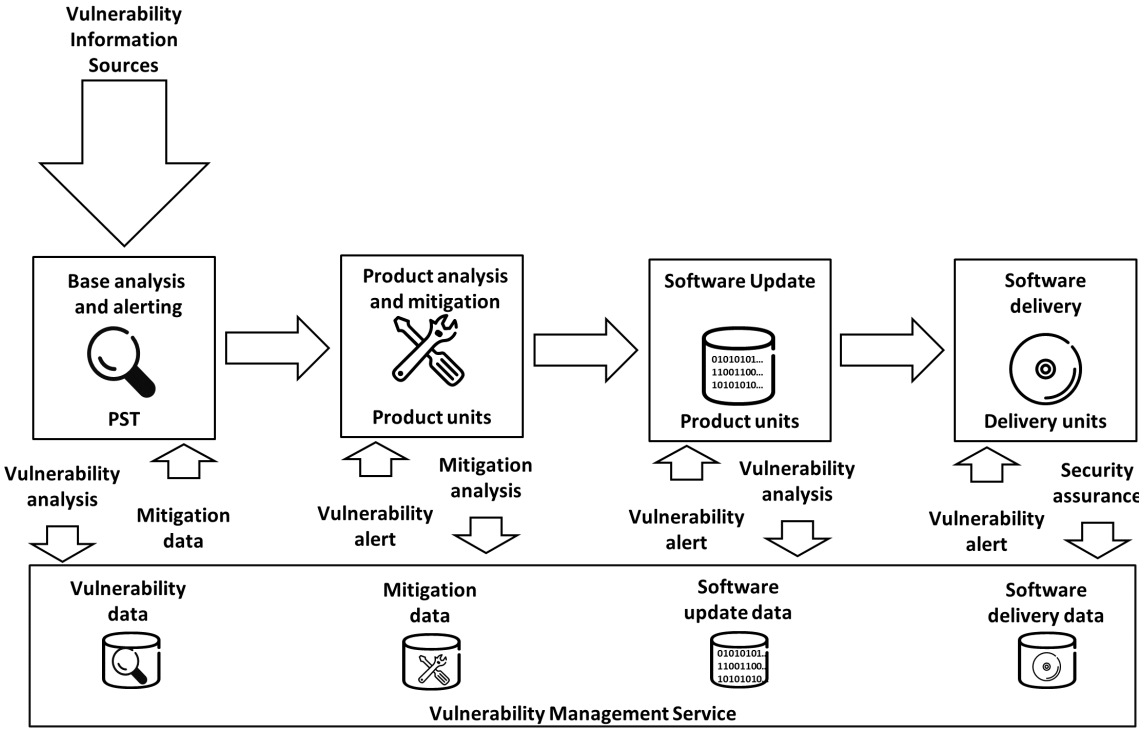

Figure 17: VMS with added product release process

During the product release, it is also recommended to implement cryptographic integrity protection for the software which is made available to customers. Digital signature is one method to implement the required cryptographic integrity protection. When such signatures are used, VMS can be updated to provide references to the signatures, thus adding additional security assurance layer. The process of software signing is out of scope for this paper.

Having presented an overview of the processes in VMS, this paper continues with maturity model for vulnerability management practices.

4.3.5 Maturity model

For the purposes of this thesis, an extension to the popular Capability Maturity Model (CMM) [51] is proposed. In this thesis, the model is used to measure maturity of vulnerability management practices.

The extension to CMM is described with the following five levels:

- Initial - The vulnerability management process is ad-hoc and could be chaotic. Not all processes are defined and execution depends on individual's effort.

- Repeatable - The minimum vulnerability management processes are in place. Discipline is necessary repeat successful execution of previous efforts.
- Defined - The vulnerability management processes for management and software engineering are documented and integrated into software development processes of the organization.
- Managed - It is possible to measure in detail the quality and performance of the vulnerability management processes. The processes are measurable and controlled.
- Optimizing - Continuous vulnerability management practices improvement is made possible with quantitative feedback from processes execution. Piloting new ideas and process enhancements.

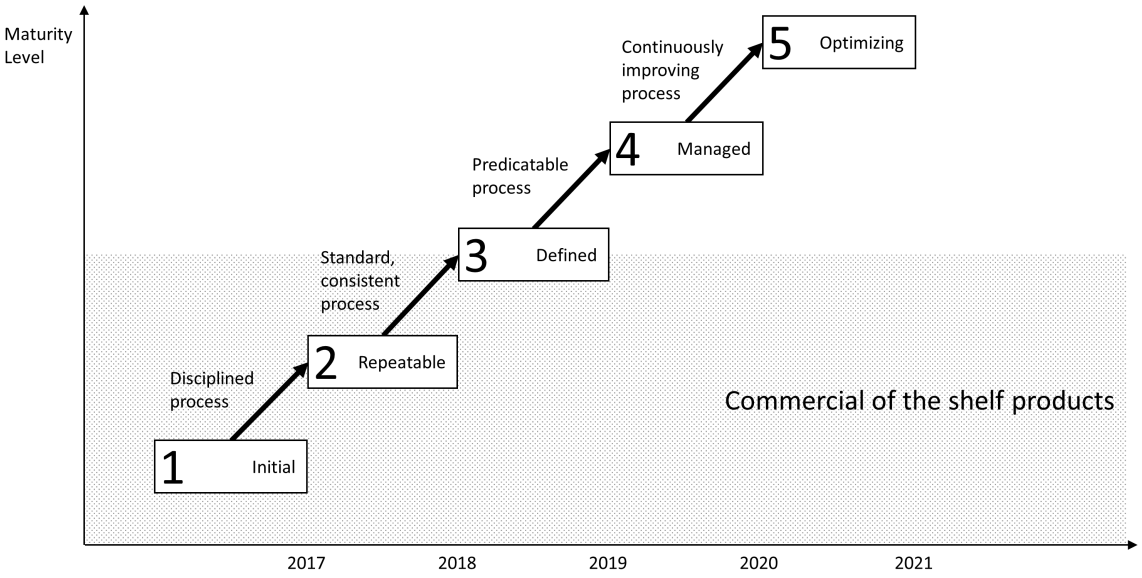

Figure 18: CMM for vulnerability management practices

Figure 18 above shows that the first level in the model is the baseline. There are some vulnerability management activities, but they are not structured and depend on the people who execute the actions. The first level is often referred to as "firefighting mode" of operations. At this stage the company should consider formalizing processes and training employees.

The second level is associated with devising strategy, writing policies and defining processes. Planing for new actions is based on previous experience. Discipline is required for successful execution of the task at hand. After these activities are completed, the company has the baseline needed to proceed with VMS deployment.

The third level is characterized with integration of existing processes in VMS. The processes are being implemented in a system to be able to deliver VMS. This could be the MVP version of VMS. In addition, this is the level which could be

reached with majority of COTS products. While COTS often implement more than MVP, they rarely provide quantitative metrics for process performance.

The forth level is defined by quantitative measurable goals for process and service performance. VMS should be deployed for the entire company portfolio and management should define performance goals. To reach this level of maturity, it is likely that VMS should be integrated with other existing PLCM machinery. The integration with other PLCM systems is one of the key aspects of this thesis. Having a stand-alone VMS, which is able to handle vulnerabilities throughout product life cycle, is an expensive and inefficient deployment scenario. It will require a much larger amount of manual work to synchronize data between all systems and it is prone to errors. To be able to meet customer requirements and performance expectations, VMS must integrate with systems from sourcing, development, delivery and support organizations.

The fifth level is focusing on optimization and continuous improvements. The company should identify flaws and weaknesses as well as proactively improve VMS. New technologies and practices can be implemented to deliver more cost effective VMS. Some of these improvements could require major service redesign and additional requirements specification. An example of such improvement is the addition of real-time threat intelligence feeds to VMS. It can be implemented as an additional vulnerability input module to increase the value and coverage of the service. Other modules can be developed to interface with customer Security Information and Event Management (SIEM) systems. In this case VMS could be used as a threat intelligence input for customer SIEM, or SIEM could feedback real-time data from live networks with information regarding vulnerability exploit attempts. These and further improvements of VMS should be considered in further studies.

Referring to the original CMM for software development, this study recommends implementing VMS to be able to reach level 4 and 5 of software quality process. With the help of VMS, a company should be able to have predictable quality of their software and ensure that it has a minimal amount of flaws on release. While VMS deployment is not mandatory in the classic CMM, it could be a valuable addition.

Following the requirements and process specifications developed as part of this thesis, an enterprise scale VMS was developed and deployed. The deployed service is described in the next section.

4.4 Enterprise Vulnerability Management Service

As an answer to the research question of this thesis, an enterprise scale VMS was developed and deployed. This section of the thesis describes the details of service acceptance testing, exploitation, benchmarks and progress in the maturity model.

4.4.1 Acceptance testings - compliance with requirements specification

A number of checklists were designed to ensure that the VMS fulfills the requirements produced as a part of the thesis. The checklists were utilized during the acceptance testing phase of the service deployment. Representatives from each VMS user group

and VMS stakeholders were present to confirm that the service in fact implements the requirements as specified. As the service was deployed in multiple releases, for practical reasons, the checklists were split to cover the expected features and functionality in each release. When the acceptance testing was completed, VMS stakeholders have signed the acceptance test report and have approved deployment of each release to production. Samples of the acceptance testing checklists are provided in Appendix B.

The requirements for VMS can be divided into two main categories: functional and non-functional. While the functional requirements focus on system features, which will be observed by the users, the non-functional requirements emphasize aspects related to operations and maintenance of the service. The requirements identified in this stage were used in the Statement of Compliance (SOC) checklist as seen in Appendix B. Table 6 provides a summary of the high level functional and non-functional requirements.

Functional requirements	**Non-functional requirements**
• User interface requirements	• Capabilities requirements
• Automation requirements	• Serviceability requirements
• Reporting requirements	• Latency requirements
• Alerting requirements	• Monitoring requirements
• Auditing requirements	• Logging requirements
• Database requirements	• Operations requirements
	• Quality requirements
	• Security requirements
	• Integrity requirements
	• Confidentiality requirements
	• Availability requirements
	• Backup requirements

Table 6: Top strategic and operational factors

It should be clear that, while non-functional requirements are not visible directly to the users, they are high priority for service stakeholders and operations team. Thus, the service shall not be deployed unless both functional and non-functional requirements are met. The progress of service development leads to a larger number

of requirements being fulfilled. When all requirements pass the acceptance tests, VMS will be considered completed and have reached the fourth level of the maturity model.

4.4.2 Exploitation phase

Once MVP version of the service is ready, it should be deployed to collect user feedback and improve it. It was decided to utilize DevOps approach. Having a team where developers and operations are tightly coupled has its benefits. There is a clear communication and developers are able to receive instant feedback from service behavior. In addition, if there is a major service malfunction, the developers are available to troubleshoot and implement the necessary measures to mitigate the problem at hand.

However, DevOps teams tend to be understaffed due to the expectation of high person utilization. When the service is functioning according to specification, the operations members will be idle. They could be utilized as developers at that time. It is also possible that developers are idle while operations are overloaded. It should be clearly communicated to all team members that service availability, confidentiality and integrity are the highest priority. Thus, operation tasks must be prioritized regardless of development release schedule or other factors.

To increase efficiency, it is possible to reuse existing company IT support and operations infrastructure. It is valuable to communicate and cooperate with existing IT teams, to ensure smooth service deployment and satisfactory user experience. While the service is focusing on product development, it is essentially an IT system and thus should follow all applicable requirements and processes.

As the company had experience with older generation vulnerability management processes, the author was able to perform series of benchmarks and compare the old system and the new service. Another series of meetings with users was held to collect their feedback and understand whether the newly deployed service has met their expectations. The results are provided in the next subsection.

4.4.3 Benchmarks and user experience

The deployment of VMS after passing the acceptance tests have ensured that the service is fulfilling the requirements. While there are possibilities for improvements, VMS is able to serve the required amount of simultaneous users. The service has been deployed in three tiers:

- Web client
- Business logic
- Database

The web client is browser based and it is built on top of REST API. If users need to improve it or automate it, they can implement their own client, based on the API documentation and libraries supplied. The business logic is protected by firewalls

and IDS/IPS. Authentication is delegated to corporate AD servers. Authorization is performed by the service itself. The database is protected with another layer of network firewalls and security appliances. The business logic and database tiers are implemented as stretched cluster for scalability and high availability. Minimum two cluster nodes are available at each location.

From performance point of view, major challenge is latency. Even with geographically redundant deployment and routing to the nearest node, physics laws cannot be circumvented. However, the service has been deployed in specific locations to ensure that majority of the users will be able to use it within the specified latency limits.

From capacity perspective, the service has met all requirements. The database and storage subsystems are performing as expected. Network bandwidth to the servers has not been a bottleneck. The service has been deployed as a cluster. If more capacity is required in the future, the cluster can be simply scaled horizontally to deliver it.

To summarize, VMS is able to handle:

- Hundreds of concurrent users
- Hundreds of alerts per second
- User interaction with latency lower than 100ms
- Unlimited* alerts stored
- Unlimited* products registered
- Unlimited* users registered
- Unlimited* components registered
- * There is actual limit which can be overcome by simply scaling the storage subsystem.

It is important to note that, as the service has been developed in-house and is being operated in-house, there should be no license fees, which would grow with service scaling up. As most of the COTS alternatives had billing based on usage or users, the custom VMS provides better cost efficiency when scaled to such size.

In addition to a number of service performance metrics, the users observed a number of quality improvements as well. VMS is able to increase visibility and enforce access restriction to information, better than older systems. This has led to better oversight of product security and improved vulnerability alert answers from product units.

Having standard identifiers for components (CPE), vulnerabilities (CVE) and scoring (CVSS), has also provided improved visibility and traceability of vulnerabilities. Users are able to perform searches with higher accuracy and VMS is able to match with improved certainty impacted products. VMS has also enforced specific fields required for vulnerability analysis. This structure has made the analysis information searchable and indexable, which increases its value.

Integration of alerting and work flow system has improved also answer rates with help of reminders. The internal notification system has allowed for improved information flow between dependent products. Whenever platform type of product provides their analysis to a vulnerability, the information is instantly available in read-only mode to all products which have registered as users of that platform.

As VMS has been deployed following standard IT processes, it has shown significant improvements by fulfilling a number of non-functional requirements. Example of such improvements are: better user support; improved backup and availability processes; as well as high availability deployment. Accounting has been improved as well by implementing RBAC and HBAC with remote secure audit logging servers. This allows also for improved traceability of user actions and faster detection of malicious activities.

To ensure that the VMS is free of known vulnerabilities, it has been registered for vulnerability monitoring in it self. This allows the DevOps team to receive vulnerability alerts and act in timely manner to protect the service. At the time of writing, VMS has not fulfilled all requirements, as it is still being developed. The following subsection describes the maturity level reached at the time of writing.

4.4.4 Service maturity

Compared to manual process, even MVP version of VMS has been a major improvement. At the time of writing, VMS has reached the third level of maturity, which have been described earlier. In practice, this means that strategy, policy and processes are defined and in place, MVP version has been deployed to provide systematic execution of the processes. VMS releases are currently focusing on UX improvements and integration with other corporate tools and systems. In parallel, the deployed VMS is being optimized and scaled as user needs require. This shows that VMS, as deployed at the time of writing, has already surpassed the maturity of a number of COTS products. This has been achieved within budget and thus has proven that developing own VMS instead of buying one was the correct decision.

Within few of years, VMS should be able to reach the fifth level of maturity. At that point of time, VMS would be close to the service depicted on the Figure 19.

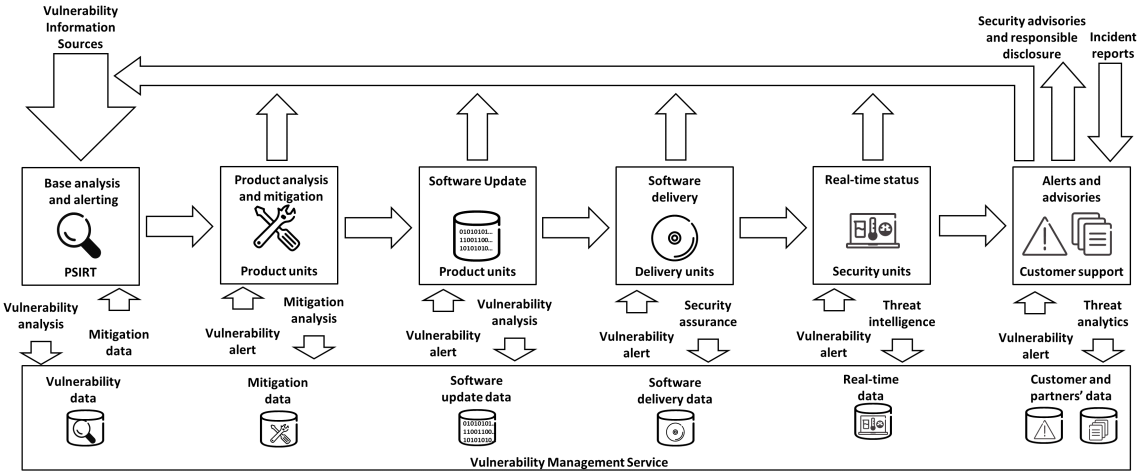

Figure 19: Next generation enterprise vulnerability management service

The addition of real-time status view and integration with customers and other partners will further increase the value of this VMS. One can say that basic principles of value network designs apply in this case as well. Having more parties connected and communicating over VMS would increase the overall value of the services and provide better value for all participants.

This concludes the Solution chapter of this thesis. The following Conclusion chapter will provide a summary of the results and suggest topics for improvements and future research.

5 Conclusion

The work on this thesis was done over the duration of more than three years. The design, development and deployment of the service took two years. The goals set by the customer were successfully achieved. As the company, that commissioned the service has over 400 products ranging from hardware to SaaS offerings - VMS was built generic by design. The modular architecture and APIs will help to integrate it and expand it with minimal effort in the future.

The concepts used within the system are defined as abstract entities and can be used recursively. User management is delegated to standard external system, which also reduces privacy impact on the system. VMS does not store users' personal data. In addition, the user permissions are defined in expandable hierarchical manner, which makes it easy to add new roles and map users to groups, which in turn will automatically grant them the permissions inherited from their group membership.

5.1 Results

The first phase of this thesis starts with research on vulnerability management systems and services. Then, it proceeds with collection of information and requirements with the help of semi-structured focus groups interviews. Once all requirements have been collected, formulated and approved by stakeholders, the study continues with evaluation of available COTS products and make-or-buy analysis.

The second phase of this thesis examines the design, development and deployment of in-house developed VMS. As VMS stakeholders decided to implement own design instead of buying COTS product, the author facilitated the design and development of the enterprise scale VMS. While the service was being developed and deployed, the author had a role to observe the progress and provide feedback with acceptance testing and future improvements. Since VMS official launch, the project has been handled by PSIRT and VMS DevOps team and the author's involvement has been brought down to a minimum.

To answer the research question, "Which is the solution able to deliver enterprise scale, cost effective and fast vulnerability management service for product life cycle?", this study concludes that for this particular customer, developing own VMS is the right approach. The interviews and benchmarks have confirmed that VMS is able to deliver more than COTS products within budget. In addition, VMS can be scaled at minimal cost of hardware and additional personnel needed to develop, operate and maintain it. The cost of VMS is not tied to usage and for enterprise customer this has proved a major financial benefit.

5.2 Assessment of results

As a result of this thesis, VMS was developed according to customer requirements. However, the thesis analysis is valuable for other companies facing the need for VMS. As the author has focused on utilizing standards and industry best practices, future research could repeat the results of this study. While the result of make-or-buy

analysis may differ, reproducing the rest of the research can still prove beneficial. Differentiating factor for make-or-buy will be the input from customer needs.

In addition, this study can be applied to any company, which develops products that are software or run software. To some extend it can be utilized by other industries, however standards and requirement for other industries might differ. A good example of industry where this study will be very important is Internet of Things. Based on the scale and speed of development in IoT, the author is convinced that VMS will be crucial for device manufacturers, system integrators and service providers.

It is also possible to utilize the same methods in more generic way to develop completely different service. Nonetheless, the value of this study is higher in the software and IT industries. Time and cost to adapt it to other applications might outweigh the benefit of reuse.

The role of the author was to collect user requirements, formalize them and then analyze the available COTS products. Then the author was responsible for providing input and recommendations for make-or-buy decision. The company decided to implement own VMS. At that stage, the author was responsible for the design of the service, its components, work-flows and logic. The author was also responsible for technology choices and data formatting. As VMS development progressed, the author was responsible for acceptance testing and implementation guidance. In addition, the author was responsible for VMS's non-functional requirements and proposals for their implementation.

As described earlier, the company regards the VMS implemented based on this study as a success. That said, the collection of requirements, design and development could have been done in a shorter time with help of stronger commitment from high level management. As vulnerability management strategy, policy and process development are prerequisites for VMS, it is beneficial if those are in place before starting analysis of VMS needs. It is also possible to speed up the development and deployment of VMS with the help of larger funding.

Having designed and deployed VMS, is it valuable to provide insights on how the service could be exploited further.

5.3 Exploitation of results

As the study analysis suggested, for enterprise scale, it is more cost effective to develop and deploy own VMS. Once the VMS is operational, it can be utilized to bring additional revenue. The service could be sold to customers, or provided as value added service for existing support contracts. As VMS is still being improved, decision whether it should be provided to customers and partners should be taken when it has reached fifth level of maturity. Providing VMS while it is not matured, might create negative perception due to higher amount of flaws or malfunctions.

The work, which was done of this project, has proven of great value to the author and the company, which has commissioned it. The study helped to identify problems with existing processes and practices. This work also helped to increase the visibility of the need for VMS. Once VMS was deployed, the quantitative and qualitative

benefits were easy to observe and measure.

A valuable approach to exploit the results of this study is to provide topics and guidance for future research, which is covered in the next subsection.

5.4 Future research

As mentioned throughout this thesis there are a number of improvements and topics for future research. Major uncertainties are: future governmental and international regulation. Future standardization work, can also bring new means to refer to vulnerabilities, to score them, or process them. Another important topic is to discover if there is a need for industry standard VMS implementation and interfaces.

For company that already has VMS, it is important to evaluate if it is still cost-effective and are there better means to receive the same service. For companies without VMS, it is important to at least evaluate their needs and make management aware of the risks which are taken by not implementing vulnerability management process. Other industries could benefit from similar services for vulnerability management or security assurance.

The author considers Open Source VMS or at least framework, pivotal for our society. Open Source projects have created enormous value for all of us, and have greatly improved software security. Open Source has become a popular way to create industry and de-facto standards. Security research is often community work. Open Source is also community work. Thus, it would be interesting to investigate why there is no community hosted and operated VMS to cover all Open Source software projects?

References

[1] J. M. Anderson. Why we need a new definition of information security. 22(4):308–313, 2003. ISSN 0167-4048. doi: 10.1016/S0167-4048(03)00407-3. URL http://www.sciencedirect.com/science/article/pii/S0167404803004073.

[2] D. Andrews. Formal methods and software development. In *Proceedings 1996 International Conference Software Engineering: Education and Practice*, pages 106–113. doi: 10.1109/SEEP.1996.533988.

[3] Anonymous. Why everything is hackable: Computer security is broken from top to bottom | the economist. The Economist, Apr. 2017. URL http://www.economist.com/news/science-and-technology/21720268-consequences-pile-up-things-are-starting-improve-computer-security.

[4] J. Balakrishnan and C. H. Cheng. The theory of constraints and the make-or-buy decision: An update and review. 41(1):40–47, 2005. ISSN 15232409. URL http://search.proquest.com.libproxy.aalto.fi/docview/235200626/abstract/C166DC9BE7C24358PQ/2.

[5] A. Bryman. *Social Research Methods*. Oxford University Press, 2015. ISBN 978-0-19-968945-3. Google-Books-ID: N2zQCgAAQBAJ.

[6] D. N. Burt, D. W. Dobler, and S. L. Starling. *World class supply management: The key to supply chain management*. McGraw-Hill/Irwin New York, NY, 2003.

[7] J. Callas, P. Corporation, L. Donnerhacke, I. GmbH, H. Finney, D. Shaw, and R. Thayer. Openpgp message format, 2007. URL https://tools.ietf.org/html/rfc4880.

[8] N. C. S. Centre. Taranis, 2015. URL https://www.ncsc.nl/english/Incident+Response/monitoring/taranis.html.

[9] M. Dearborn. Ford invests in pivotal to accelerate cloud-based software development; new labs drive ford smart mobility innovation, 2016. URL https://media.ford.com/content/fordmedia/fna/us/en/news/2016/05/05/ford-invests-in-pivotal.html.

[10] T. Dierks, Independent, E. Rescorla, and R. Inc. The transport layer security (tls) protocol version 1.2, 2008. URL https://tools.ietf.org/html/rfc5246.

[11] J. Dittmer. Applying lessons learned for the next generation vulnerability management system. *SANS Institute InfoSec Reading Room*, 2015. URL https://www.sans.org/reading-room/whitepapers/threats/applying-lessons-learned-generation-vulnerability-management-system-35997.

[12] FIRST. Common vulnerability scoring system v3.0: Specification document, 2015. URL https://www.first.org/cvss/specification-document.

[13] Flexera. Vulnerability intelligence manager, 2016. URL https://www.flexerasoftware.com/enterprise/products/software-vulnerability-management/vulnerability-intelligence-manager/.

[14] Flexera. Vulnerability review 2017, 2017. URL https://resources.flexerasoftware.com/web/pdf/Research-SVM-Vulnerability-Review-2017.pdf.

[15] C. for Internet Security. Cis benchmakrs, 2017. URL https://www.cisecurity.org/cis-benchmarks/.

[16] P. Foreman. *Vulnerability Management*. CRC Press, 2009. ISBN 978-1-4398-0151-2. Google-Books-ID: 4JEIoVBHdwsC.

[17] Gartner. Improve it security with vulnerability management, 2005. URL https://www.gartner.com/doc/480703.

[18] A. Greenberg. Hackers remotely kill a jeep on the highway - with me in it, 2015. URL https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[19] M. Hall. Why people are key to cyber-security. 2016(6):9–10, 2016. ISSN 1353-4858. doi: 10.1016/S1353-4858(16)30057-5. URL http://www.sciencedirect.com/science/article/pii/S1353485816300575.

[20] R. Housley and V. Security. Cryptographic message syntax (cms), 2009. URL https://tools.ietf.org/html/rfc5652.

[21] J. Humble. The case for continuous delivery, 2014. URL https://www.thoughtworks.com/insights/blog/case-continuous-delivery.

[22] M. Humphries. D'oh! 2016's biggest tech fails, 2016. URL http://www.pcmag.com/feature/350413/d-oh-2016-s-biggest-tech-fails.

[23] ICASI. The common vulnerability reporting framework, 2012. URL http://www.icasi.org/cvrf/.

[24] ISO/IEC. Information technology – security techniques – vulnerability handling processes, 2013. URL https://www.iso.org/standard/53231.html.

[25] ISO/IEC. Information technology – security techniques – vulnerability disclosure, 2014. URL https://www.iso.org/standard/45170.html.

[26] ISO/IEC. Information technology – security techniques – information security management systems – overview and vocabulary, 2016. URL https://www.iso.org/standard/66435.html.

[27] J. Jenkins. Velocity 2011, 2011. URL https://www.youtube.com/watch?v=dxk8b9rSKOo.

[28] S. Keach. 12 of the biggest tech fails from 2016: Can you guess the worst?, 2016. URL http://www.trustedreviews.com/news/biggest-tech-fails-2016.

[29] B. Krebs. Hacked cameras, dvrs powered today's massive internet outage, 2016. URL https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/.

[30] B. Krebs. Krebsonsecurity hit with record ddos, 2016. URL https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[31] S. MacConnell. *Code complete: a practical handbook of software construction*. Microsoft Press, 1993.

[32] M. Michael. The state of cyber security 2017, 2017. URL https://business.f-secure.com/the-state-of-cyber-security-2017.

[33] T. T. Mikko Hyppönen. F-secure state of cyber security 2017. techreport, F-Secure, Feb. 2017. URL http://images.news.f-secure.com/Web/FSecure/%7Bd52f77ef-dd23-4871-ab9b-2ae794f4dadd%7D_F-Secure-Threat-Report-State_of_Cyber_Security_2017.pdf.

[34] Mitre. A community-developed list of software weakness types, 2007. URL https://cwe.mitre.org/documents/vuln-trends/index.html.

[35] Mitre. Common platform enumeration: A structured naming scheme for it systems, platforms and packages, 2011. URL https://cpe.mitre.org/specification/.

[36] Mitre. Common vulnerabilities and exposures the standard for information security vulnerability names, 2014. URL https://cve.mitre.org/.

[37] Mitre. Open vulnerability and assessment language: A community-developed language for determining vulnerability and configuration issues on computer systems, 2014. URL http://oval.mitre.org/.

[38] Nessus. Vulnerability management, 2016. URL https://www.tenable.com/products/tenable-io/vulnerability-management.

[39] Nessus. Vulnerability management and analytics, 2017. URL https://www.tenable.com/products/securitycenter.

[40] NIST. Standards, 2001. URL http://csrc.nist.gov/groups/STM/cmvp/standards.html.

[41] NIST. Product lifecycle management support: A challenge in supporting product design and manufacturing in a networked economy, 2005. URL http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=822275.

[42] NIST. Scap specification, 2011. URL https://scap.nist.gov/revision/index.html.

[43] NIST. Guide for conducting risk assessments, 2012. URL http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-33.pdf.

[44] NIST. Official common platform enumeration (cpe) dictionary statistics, 2017. URL https://nvd.nist.gov/products/cpe/statistics.

[45] NIST. Vulnerability statistics, 2017. URL https://nvd.nist.gov/vuln/search/statistics?adv_search=false&form_type=basic&results_type=statistics&search_type=all.

[46] NOPSEC. Total cost of ownership for vulnerability management, 2014. URL https://www.nopsec.com/blog/total-cost-ownership-vulnerability-management/.

[47] OASIS. Oasis commong security advisory fremework (csaf) standard work, 2016. URL https://cvrf.github.io/index.html.

[48] OWASP. Open web application security project, 2017. URL https://www.owasp.org/index.php/Main_Page.

[49] OWASP. Owasp top ten project, 2017. URL https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[50] I. Paul. The year in tech: 2016's biggest flops, fails, and faux pas, 2016. URL www.pcworld.com/article/3152054/internet/the-year-in-tech-2016s-biggest-flops-fails-and-faux-pas.html.

[51] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber. Capability maturity model, version 1.1. 10(4):18–27, 1993. ISSN 0740-7459. doi: 10.1109/52.219617.

[52] PCI. Maintaining payment security, 2017. URL https://www.pcisecuritystandards.org/pci_security/maintaining_payment_security.

[53] Qualys. Best practices for selecting a vulnerability management (vm) solution, 2013. URL https://www.qualys.com/docs/whitepapers/best-practices-selecting-vulnerability-management-solution.pdf.

[54] Qualys. Vulnerability management, 2017. URL https://www.qualys.com/suite/vulnerability-management/.

[55] B. Ramsdell, B. S. Labs, S. Turne, and IECA. Secure/multipurpose internet mail extensions (s/mime) version 3.2 certificate handling, 2010. URL https://tools.ietf.org/html/rfc5750.

[56] B. Ramsdell, B. S. Labs, S. Turner, and IECA. Secure/multipurpose internet mail extensions (s/mime) version 3.2 message specification, 2010. URL https://tools.ietf.org/html/rfc5751.

[57] Rapid7. On premises vulnerability scanner, 2015. URL https://www.rapid7.com/products/nexpose.

[58] Rapid7. Live vulnerability management and endpoint analytics, 2017. URL https://www.rapid7.com/products/insightvm/.

[59] D. Reisinger. 13 biggest security fails of 2016, 2016. URL http://www.tomsguide.com/us/pictures-story/986-years-security-fails.html.

[60] S. S. Response. What you need to know about the wannacry ransomware, 2017. URL https://www.symantec.com/connect/blogs/what-you-need-know-about-wannacry-ransomware.

[61] F. Richter. 200,000+ systems affected by wannacry ransom attack, 2017. URL https://www.statista.com/chart/9399/wannacry-cyber-attack-in-numbers/.

[62] RSA. Rsa archer it & security risk management, 2016. URL https://www.rsa.com/content/dam/rsa/PDF/2016/05/h15021-rsa-archer-itsrm-sb.pdf.

[63] I. Sean Turner. Using sha2 algorithms with cryptographic message syntax, 2010. URL https://tools.ietf.org/html/rfc5754.

[64] D. Shephard. 84 fascinating & scary it security statistics, 2015. URL https://www.netiq.com/communities/cool-solutions/netiq-views/84-fascinating-it-security-statistics/.

[65] T. Spangler. The biggest tech fails of 2016, 2016. URL http://variety.com/2016/digital/news/2016-biggest-tech-fails-1201945122/.

[66] J. Stark. Product lifecycle management. In *Product Lifecycle Management*, pages 1–29. Springer, 2015.

[67] Symantec. Internet security threat report, Apr. 2016. URL https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf.

[68] Symantec. Internet security threat report, 2016. URL https://www.symantec.com/security-center/threat-report.

[69] Symantec. Mirai: what you need to know about the botnet behind recent major ddos attacks, 2016. URL https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks.

[70] E. Tittel. Seven criteria for buying vulnerability management tools, 2016. URL http://searchsecurity.techtarget.com/feature/Seven-criteria-for-buying-vulnerability-management-tools.

[71] U.S.Department. Summary of the hipaa security rule, 2017. URL https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/.

[72] A. Vault. Unified security management, 2017. URL https://www.alienvault.com/products.

[73] Veracode. Vulnerability management platform and programs., 2017. URL https://www.veracode.com/security/vulnerability-management.

[74] VulnDB. Vulnerability quickview 2016 year end. Technical report, Risk Based Security Inc., Jan. 2017. URL https://pages.riskbasedsecurity.com/hubfs/Reports/2016VulnDBYearEndReport1.27.17.pdf.

[75] S. M. Welberg. Vulnerability management tools for COTS software-a comparison. URL http://doc.utwente.nl/64654. 00009.

[76] J. D. Wisner, K.-C. Tan, and G. K. Leong. *Principles of supply chain management: A balanced approach*. Cengage Learning, 2014.

[77] D. Yadron and D. Tynan. Tesla driver dies in first fatal crash while using autopilot mode, 2016. URL https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk.

A Appendix

Table 2: Tool comparison		Platform			Magni -tude		Standards					Type of vul- nerabilities			Analysis				Types of results		info
Vendor	*Name*	*Windows*	*Linux*	*Other*	*single*	*enterprise*	*XCCDF*	*CVE*	*CPE*	*CCE*	*CVSS*	*configuration*	*source code*	*environment*	*Compliance checking*	*Patch Management*	*Vulnerability scanning*	*Correlated analysis*	*Quantitative*	*Qualitative*	*Information available*
Tenable	Nessus [54]	X	X	X		X	X	X			X	X	X	?	X	X	X		X	X	X
	Security Center 3 [55]	X	X	X		X	X	X			X	X	X	X	X	X	X	X	X	X	X
GFI	LANguard Vulnerability manager 8 [21]	X	X	X		X		X				X	X	?	X	X	X			X	X
Secure elements	C5 Platform [47]	X	X	X		X	X	X	X	X	X	X	X	?	X	X	X		X	X	X
Threatguard	Secutor Prime Free [57]	X			X		X	X	X	X	X	X		?	X	X	X		X	X	
	Secutor prime Magnus [58]	X				X	X	X	X	X	X	X		?	X	X	X		X	X	
	Vulnerability Management System [59]	X	X	X		X	X	X	X	X	X	X	X	?	X	X	X		X	?	
Belarc	Belarc NIST Advisor [8]	X			X			X		X	X	X		?	X	X			X		
IBM	Internet Scanner [24]	X	X	X		X		X				X	X	?		X	X			X	
	Tivoli Security compliance manager [23]	X	X	X		X	?	?	?	?	?	X	?	?	X	X	?	?		X	
CA	Vulnerability manager r8.3 [13]	X	X	X		X	?	?	?	?	?	X	X	?	X	X	X		?	?	
Qualys	QualysGuard Enterprise [42]	X	X	X		X		X			X	X	X	?	X	X	X		X	X	X
Skybox	Secure [52]	X	X	X		X		X			X	X	X	?	X	X	X	X	X	X	
Amenaza	Secure*IT*ree [3]	X	X	X		X												X	X		X
Gideon Technologies	SecureFusion Portal [22]	X	X	X		X	X	X	X	X		X	X	?	X	X	X			X	
CIS	CIS-CAT [14]	X	X	X	X		X	?	?	?	?	X	?	?	X	X	?	?	?	?	
Configuresoft	Enterprise Configuration Manager (ECM) [16]	X	X	X		X	?	?	?	?	?	X		?	X	X			X		
Rapid7	NeXpose Unified vulnerability management (UVM) [43]	X	X	X		X		X				X	X	?	X	X	X		X	X	X
Core Security	Core Impact 6.0 [17]	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	
eEye	Retina Network Security Scanner [18]	X	X	X		X		X				X	X	?	X	X	X			X	X
McAfee	Policy Auditor and Remediation Manager (PARM) [31]	X	X	X		X		X		X		X		?	X	X			?	?	
	Foundstone [30]	X	X	X		X	?	?	?	?	?	X	X	?	X	X	X			X	
NetIQ	Risk and Compliance Center [38]	X	X	X		X		X			X	X	X	?	X	X	X		X	X	X
nCircle	Configuration Compliance Manager [37]	X	X	X		X		X				X		?	X					X	X
Shavlik	NetChk Compliance [49]	?	?	?		X	?	?	?	?	?	X		?	X				?	?	
	ARM [48]	X	?	?		X	?	?	?	?	?	X	X	?		X			?	?	
Microsoft	Microsoft baseline security analyser [32]	X				X						X		?		X			X		
Cisco	Cisco IntelliShield Alert manager [15]	X				X		X			X	X		?	X		X		X		
Lumension Security	Patchlink Scan [26]	X	X	X		X		X				X	X	?	X	X	X			X	X
BgFix	Discovery 7 [9]	X	X	X		X	?	?	?	?	?	X	X	?	X	X			?	?	

Figure A1: COTS product comparison [75]

B Appendix

Sample Statement of Compliance checklists for EVMS

B.1 Functional requirements

B.1.1 User Interface

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-UI-A	1	Yes	Yes	Yes	Success
Req-UI-B	2	No	10%	No	Fail
Req-UI-C	3	Yes	No	No	Fail

Figure B1: Table with user interface SOC

B.1.2 Automation

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-AT-A	1	Yes	Yes	Yes	Success
Req-AT-B	2	No	10%	No	Fail
Req-AT-C	3	No	No	No	N/A

Figure B2: Table with Automation SOC

B.1.3 Reporting

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Rep-A	1	Yes	Yes	Yes	Success
Req-Rep-B	2	Yes	10%	No	Fail
Req-Rep-C	3	Yes	20%	No	Fail

Figure B3: Table with Reporting SOC

B.1.4 Alerting

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Alert-A	1	Yes	Yes	Yes	Success
Req-Alert-B	2	Yes	30%	No	Fail
Req-Alert-C	3	Yes	40%	No	Fail

Figure B4: Table with Alerting SOC

B.1.5 Auditing

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Audit-A	1	Yes	Yes	Yes	Success
Req-Audit-B	2	Yes	60%	No	Fail
Req-Audit-C	3	Yes	70%	No	Fail

Figure B5: Table with Auditing SOC

B.1.6 Database

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-DB-A	1	Yes	Yes	Yes	Success
Req-DB-B	2	Yes	Yes	Yes	Success
Req-DB-C	3	Yes	Yes	Yes	Fail

Figure B6: Table with Database SOC

B.2 Non-functional requirements

B.2.1 Capabilities

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Cap-A	1	Yes	Yes	Yes	Success
Req-Cap-B	2	Yes	Yes	Yes	Success
Req-Cap-C	3	Yes	Yes	Yes	Success

Figure B7: Table with Capabilities SOC

B.2.2 Serviceability

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Serv-A	1	Yes	Yes	Yes	Success
Req-Serv-B	2	Yes	Yes	Yes	Success
Req-Serv-C	3	Yes	Yes	Yes	Fail

Figure B8: Table with Serviceability SOC

B.2.3 Latency

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Lat-A	1	Yes	Yes	Yes	Success
Req-Lat-B	2	Yes	Yes	Yes	Success
Req-Lat-C	3	Yes	Yes	Yes	Fail

Figure B9: Table with Latency SOC

B.2.4 Monitoring

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Mon-A	1	Yes	Yes	Yes	Success
Req-Mon-B	2	Yes	Yes	Yes	Success
Req-Mon-C	3	Yes	Yes	Yes	Fail

Figure B10: Table with Monitoring SOC

B.2.5 Logging

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Log-A	1	Yes	Yes	Yes	Success
Req-Log-B	2	Yes	Yes	Yes	Success
Req-Log-C	3	Yes	Yes	Yes	Fail

Figure B11: Table with Logging SOC

B.2.6 Operations

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Ope-A	1	Yes	Yes	Yes	Success
Req-Ope-B	2	Yes	Yes	Yes	Success
Req-Ope-C	3	Yes	Yes	Yes	Fail

Figure B12: Table with Operations SOC

B.2.7 Quality

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Qual-A	1	Yes	Yes	Yes	Success
Req-Qual-B	2	Yes	Yes	Yes	Success
Req-Qual-C	3	Yes	Yes	Yes	Fail

Figure B13: Table with Quality SOC

B.2.8 Security

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Sec-A	1	Yes	Yes	Yes	Success
Req-Sec-B	2	Yes	Yes	Yes	Success
Req-Sec-C	3	Yes	Yes	Yes	Fail

Figure B14: Table with Security SOC

B.2.9 Integrity

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Int-A	1	Yes	Yes	Yes	Success
Req-Int-B	2	Yes	Yes	Yes	Success
Req-Int-C	3	Yes	Yes	Yes	Fail

Figure B15: Table with Integrity SOC

B.2.10 Confidentiality

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Confid-A	1	Yes	Yes	Yes	Success
Req-Confid-B	2	Yes	Yes	Yes	Success
Req-Confid-C	3	Yes	Yes	Yes	Fail

Figure B16: Table with Confidentiality SOC

B.2.11 Availability

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Avail-A	1	Yes	Yes	Yes	Success
Req-Avail-B	2	Yes	Yes	Yes	Success
Req-Avail-C	3	Yes	Yes	Yes	Fail

Figure B17: Table with Availability SOC

B.2.12 Backup

Requirement (RQ)	RQ No.	Comply	Complete	Test	Test Result
Req-Back-A	1	Yes	Yes	Yes	Success
Req-Back-B	2	Yes	Yes	Yes	Success
Req-Back-C	3	Yes	Yes	Yes	Fail

Figure B18: Table with Backup SOC